THE DESIGN AND CONSTRUCTION OF A COMPUTER SIMULATION

COLIN P. HAMMON

THE DESIGN AND CONSTRUCTION OF A COMPUTER SIMULATION

* * * * *

Colin P. Hammon

THE DESIGN AND CONSTRUCTION OF A COMPUTER SIMULATION

by
Colin P. Hammon
Lieutenant, United States Navy

Submitted in partial fulfillment of
the requirements for the degree of

MASTER OF SCIENCE
IN
OPERATIONS RESEARCH

United States Naval Postgraduate School
Monterey, California
1964

NPS ARCHIVE
1964
HAMMOND, C.

THE DESIGN AND CONSTRUCTION OF A COMPUTER SIMULATION

by

Colin P. Hammon

This work is accepted as fulfilling
the thesis requirements for the degree of

MASTER OF SCIENCE
IN
OPERATIONS RESEARCH
from the
United States Naval Postgraduate School

ABSTRACT

A procedural approach to the design and implementation of a computer simulation model is presented, with a simulation model description and computer code. The discussion and example are intended as reference material for the computer science and computer war gaming courses offered at the United States Naval Postgraduate School. The sample computer simulation was designed for the statistical analysis of the comparative effectiveness of different ASW helicopter search tactics in a variety of tactical and physical environments. This simulation has available a wide range of input parameters and is applicable to all ASW helicopters and any search plan employing ten helicopters or less. The accompanying FORTRAN computer code was written for the CDC 1604 computer, and is adaptable to the IBM 7090-94 by the inclusion of the appropriate control cards and random number generator.

ACKNOWLEDGMENTS

The author wishes to express his appreciation to Professor Alvin F. Andrus for his encouragement and guidance during the preparation of this thesis. Thanks are also due the following for their substantial contributions to the formulation and preparation of the text or simulation: Richard L. Klinkner of the Planning Analysis Group, Johns Hopkins University, who contributed the sonar range computation procedure used in the helicopter search simulation; Robert Jenkins also of the Planning Analysis Group, who suggested the subject of the simulation; and Professor Rex H. Shudde, of the department of Operations Research, United States Naval Postgraduate School, who read and commented on the manuscript.

TABLE OF CONTENTS

CHAPTER I

INTRODUCTION

1.1 Objective.

The objective of this thesis is to offer the beginning student of computer war game simulation an easily understood example of what a computer simulation is and how one might go about constructing one. The first year Operations Analysis students of the U. S. Naval Postgraduate School constitute the particular audience for which the thesis is intended. Accordingly, no more than a basic understanding of probability and the Monte Carlo Method is presupposed. Fundamentals of War Gaming by F. J. McHugh [1] and A Survey of Historical Developments in War Gaming by John Young [2] are recommended to the reader who has not studied manual war gaming methods or is unfamiliar with the historical development of war games.

The particular method chosen by the author to reach the stated objective consists of a sample computer simulation and a proposed plan for designing and implementing a computer simulation model. The proposed simulation design methodology is essentially one solution to the major problems confronting a novice computer simulation designer.

The documentation and computer code for a helicopter anti-submarine warfare search simulation are included in Chapter VI and Appendix A respectively. This simulation, subsequently referred to as AHS-1, is cited as an example throughout this thesis. AHS-1 was written as an example because

it treats an operation of general interest in the U. S. Navy, i.e., a search for an evading submarine, and is of limited enough scope to be easily understood. This particular subject was selected as a result of the author's past experience as an Anti-Submarine Warfare Officer in helicopter squadrons. AHS-1 is intended as an analytic tool for the evaluation of the relative effectiveness of different helicopter search plans in similar environmental and tactical circumstances. It is hoped that units within the U. S. Navy, whose responsibility it is to conduct such analysis, will consider using the simulation; either as written or with any necessary modifications. With this application in mind it should be noted that a hypothetical distribution of helicopter figure of merit was incorporated in the simulation. This allowed the thesis to be unclassified. Rewriting this particular portion of the coding is a relatively minor modification.

1.2 Definitions.

Some confusion exists throughout the literature in the definition of terms used in war gaming and simulation in general. This is perhaps partly attributable to the diverse backgrounds of the individuals engaged in the field. War gamers include people from nearly every branch of science and engineering as well as professional military men. This confusion is possibly also due in part to a tendency to attach special meaning to certain terms according to the particular activity in which the individual is engaged. The following definitions will be adhered to in the remainder of this thesis and were chosen for simplicity as well as generality.

War Game has been applied to activities ranging in complexity from two opponents at a sand table to strike exercises involving thousands of men and millions of dollars worth of equipment. A. W. Pennington chooses to leave the term undefined because of the almost universal disagreement on a definition. [3] F. J. McHugh defines a war game as "a simulation, in accordance with predetermined rules, data, and procedures, of selected aspects of a conflict situation." [1] This definition is favored by the author because it is clear and concise and does not tend to imply any special type of war game, size or mode of play. The words "conflict situation" should be particularly noted. This is the primary distinction between a game and other types of simulation. In accordance with current usage, war game will not include exercises in which real forces or equipment are employed.

Because of its size and complexity, the real world, or even selected aspects of the real world, is usually impossible to study in detail in the laboratory. Consequently, scientists must usually resort to the study of a model of a real life situation. This model may be iconic, such as a model airfoil constructed by the aeronautical engineer for study in a wind tunnel, or it may be symbolic. A stick and gumdrop model of an atom is a symbolic model, as is the familiar equation $F = ma$. The latter is an example of a mathematical model and together with several other equations, or models, is the model for Newtonian mechanics. In essence, a model is a representation of some aspect of real life on which experiments can be performed; and from the behavior of the model certain laws or

general rules governing the real world situation may be inferred. It is characteristic of every model that the resulting conclusions about the real world will never be any more valid than the model itself. This thesis will be concerned with mathematical models.

Mathematical models can be further classified as deterministic or stochastic. The equation F = ma is a <u>deterministic model</u> in that if mass and acceleration are known the force is uniquely determined. If, however, the measurement of acceleration is subject to certain random errors the above equation could be written as

$$F = m(a+e)$$

where a is the measured value of acceleration and e represents a random measurement error. In this example, F is a random variable, and the equation is a <u>stochastic model</u> representing an abstract concept which is called force. To illustrate, the movement of the target submarine in AHS-1 during any time interval is represented by equations involving the course and speed of the submarine during this increment of time. Given course, speed, and time of travel, the distance traversed and the direction can be predicted exactly. The course itself though, is assumed to be a random variable distributed uniformly on the interval $[0,360)$ degrees.

$$COURSE = (000+RN)$$

where RN is a random number drawn from a set of numbers distributed uniformly between zero and 360. Therefore, the direction traveled during the time interval is a random variable and the model is stochastic. Once the random number has been

drawn, COURSE becomes the observed value of a random variable and the direction of travel is uniquely determined.

The word simulation is used interchangeably with model by many writers in the field. We choose to define a distinction between the two. As used in this paper, a simulation includes both a model and its logic. Consider the model as a plan which is essentially static until put into effect. The simulation is the dynamic implementation of this plan, and includes the program logic and the completed computer instructions. When the term simulation is used in this thesis, computer simulation will be implied unless the context makes it clear that this is not the case. A computer war game is one particular type of simulation.

A. S. Householder defines the Monte Carlo method as "... the device of studying an artificial stochastic model of a physical or mathematical process..." [4] The term Monte Carlo is relatively new but the method is the technique of model sampling used by statisticians for at least sixty years. An alternative and perhaps simpler statement of the above definition is "the use of sampling to estimate the answer to a mathematical problem." [5] For a comprehensive treatment of Monte Carlo techniques and applications the reader is referred to Symposium on Monte Carlo Methods, edited by H. A. Meyer. [6]

The terms real world and system will be used throughout the thesis and in this particular context are considered to be synonymous. Both terms are used to describe the particular part of the universe which is being studied. The real world, or system, could comprise the particles existing in the nucleus

of an atom, or it might consist of a portion of the ocean, a submerged submarine and a flight of sonar equipped helicopters. In the latter example the weather, sonar operators and pilots, or anything else which might interact with the helicopters or submarine during a search operation would be part of the system.

CHAPTER II

COMPUTER SIMULATION DESIGN

The first principle of computer simulation design is that every design situation is unique. Each simulation presents problems which must be solved on the basis of the objectives of the particular study and the characteristics of the real world system under consideration. It is believed, however, that certain factors of a general nature must be considered in any computer simulation design and that there is a logical order in which these factors should ordinarily be examined. The purpose of this chapter is to describe a general methodologic approach to computer simulation design.

Before attempting to describe a methodology of general applicability to computer simulation the question of why such a procedure is needed should be answered. War gaming has a long history of application to military problems and in the course of this history many principles of game design have evolved and been recorded. Unless there are some basic differences in the objectives as well as techniques associated with the design process, these principles of classical war game design should be equally applicable to computer simulation. In an attempt to determine if such differences do exist, some of the factors contributing to the philosophy of both classical war game and computer simulation design will be considered.

2.1 Effect of the training role on classical war game design.

War games began to achieve the stature of a serious business rather than a form of amusement in the early 19th century. [2,7] War games from this time on were for the most part extensions of the field maneuver. Field maneuvers provided a means of practicing for war without the high costs in life and resources associated with actual combat. Similarly war games offered militarists as well as prospective military officers the opportunity to practice the business of war without fielding troops and their associated equipment.

In both field maneuvers and manual war games, the primary purpose is generally training of the participants rather than analysis of the effects of factor interactions on the outcome. Because people tend to learn more rapidly when placed in familiar surroundings, the game designer attempts to create a simulated environment which will be familiar to the players. As a result, game factors are chosen because of their relation to the appearance of reality rather than the effect these factors might or might not have on the final outcome of the game.

If general trends or cause-and-effect relationships between numerical or qualitative inputs and game results are noted, they are often considered as a secondary return rather than a primary game objective. Consequently the analysis of results, and the question of what specific factors to analyze, is generally not considered a part of game design, and is often left entirely to the decision maker or his staff. The nature of this analysis is also related to the role of war gaming as a training device. Throughout the play of the game

a detailed battle history is usually maintained. This history is then studied from the standpoint of the effect of each individual decision on the final outcome of the game. If the game is repeated, the players and decision rules may change. In this way the participants are offered the opportunity to react to a wider range of stimuli. This is desirable from an educational standpoint. However, when war games are used for analytic purposes this type of game may test the player rather than operational plans or doctrines.

2.2 Effect of digital computers on game design.

The digital computer has contributed more than computational speed to simulation. With this means of rapid calculation the analyst has gained a degree of control and repeatability not attainable in manual war gaming. Unlike manual gaming in which decision rules and players might change between plays, these factors do not vary between replications of a computer simulation. Neither will the computer simulation results be affected by the participants learning the game nor by some unusually bold stroke of genius exhibited by one of the players. Through the elimination of human intervention, a major source of uncontrollable experimental error, and the gain in repeatability, computer simulation offers the analyst an analytic rather than a primarily educational tool.

The automation of war gaming has also placed certain restraints on the simulation designer and these restraints must be reflected in any plan for simulation design. The amount of information which can be stored in the memory of a computer

is limited and computer running time is expensive, on the order of ten dollars a minute for the IBM 7094. Furthermore, before the simulation can be run on a machine it must be reduced to a set of computer instructions. This list of instructions, or computer program, may represent a considerable investment in time and money. Large computer simulations such as the Air Battle Model described by R. H. Adams and J. L. Jenkins [8] may be two years or more in the construction. [9] Because of these inherent limitations, it is usually necessary to exclude certain factors which do not contribute substantially to the purpose of the game and may even detract from it by masking the game response to factors which do. The problem confronting the simulation designer is not just how to achieve a high fidelity representation of the real world, but how to selectively choose those components of the real world which characterize the particular questions to be answered. A major effect of digital computers on game design is that a need has arisen for criteria which will help the designer in deciding what is and is not pertinent to the game purpose.

2.3 Objectives of a computer design methodology.

The preceding discussion suggests that computer war gaming entails something more than adapting manual games for play on high speed computing machines. It seems reasonable then that computer game design should embody more than a straightforward extension of manual war game techniques. The foregoing also indicates some of the requirements a plan for simulation design should satisfy. These are:

(1) The simulation should be designed with specific questions in mind.

(2) The design should capitalize on the advantages of repeatability and precise parameter control offered by high speed computers. This implies utilizing some means of reducing the effect of experimental error.

(3) Emphasis should be placed on diminishing the disadvantages of limited space and high cost associated with digital computers.

(4) Factors should be chosen on the basis of their relation to simulation objectives.

(5) Planning of the analysis and interpretation of results should be concurrent with designing the simulation.

2.4 Experimental design approach to simulation.

It is considered that the above requirements are best met by the methods of experimental design. This is not surprising since a simulation is basically a sampling experiment. The principle departure of simulation from the agricultural experiments, which gave experimental design its impetus, is founded in technique and degree of abstraction rather than theory. In simulation the experimentor first builds a representation or model of the real world and then performs his experimentation on this model. The agricultural worker may use real plants grown in real soil as the basis of his experiment, but not just any plants grown on just any soil. The plants are carefully chosen and plots are arranged in a selected pattern in order to control the effect of extraneous factors such as soil variability or differences in drainage. Similarly, the simulation designer selects factors according to whether they contribute to the essence of realism in a

specific situation. The agricultural experimentor eliminates weeds which might, by taking water from the plants, result in an erroneous estimate of the characteristic yield. A problem confronting the simulation designer is that he must first determine which are "plants" and which "weeds". Once this determination has been made he must resist the temptation to simulate the "weeds" just because they are part of the real world he is dealing with.

A further motivation for the use of the experimental design approach is suggested by I. F. Wagner.

> There have been times when the inexperienced War Gaming researcher has set up his own study, planned his own runs and after the runs have been made, offered his data for someone to interpret. The result is usually an inability to place proper interpretation upon the results, necessary runs have been omitted, redundant and superfluous runs have been conducted and the time and money expended may exceed the value received. [10]

Implicit in this approach is that selection of factors to include in the model is closely related to data collection. The collection of data is as important a part of statistical analysis as data preparation and interpretation.

For those who may not be familiar with basic experimental design the following from Design and Analysis of Industrial Experiments, edited by O. L. Davies, should further clarify the relation between experimental design and the discussion to follow.

> The first step in planning any experiment is to form a clear picture of objects of the experiment, the factors which may affect the results, and the errors which will inevitably arise. The chief objects of the experimental design are:
>
> (i) To arrange the experiment so that the effect of changing each condition can be readily measured and separated from the effects of changing the

other conditions, and from experimental error.

(ii) To obtain a valid estimate of error appropriate for assessing the statistical significance of the effects of the factors.

(iii) To enable the effects to be measured with the required accuracy. [11]

The preceding principle of experimental design will play an important role in determining the course to be followed in designing a simulation. The first step will be to form a clear picture of the objects of the experiment and a first approximation of the factors affecting the results. This is treated in Chapter III, and will take the form of a preliminary study of the simulation purpose and the system to be simulated.

Consideration of the sources of error and the first objective of experimental design stated above will shape the main body of the proposed simulation design method. The necessity of designing the simulation so that the effects of varying parameters can be measured and separated from the effects of errors and each other was seen to be a basic departure from classical war game requirements. This objective will be realized in the computer simulation by designing the model itself as a factorial experiment, and is the subject of Chapter IV. The second and third objectives of experimental design are intrinsic to model testing, and will be discussed in conjunction with implementation of the model in Chapter V.

Before proceeding to the preliminary study phase a brief discussion of factorial experiments is in order. This class of experimental designs wil constitute the pattern for the

simulation model.

2.5 Factorial experiments.

Factorial experiments represent a class of experiments which are particularly applicable to studying variation deliberately introduced by the experimentor. [11,12] The specific techniques associated with constructing these experiments is beyond the scope of this thesis and will not be considered. The intention here is to consider certain very basic concepts of experimental design rather than techniques. The purpose of this discussion is to indicate how these concepts can help the simulation model designer in selecting inputs which reflect the objectives of the system as they pertain to the simulation purpose.

The following terms will be used throughout the remainder of the discussion of factorial experiments and simulation design.

Factor. In general a factor is any quantity which contributes to the determination of the state of the simulated system, and can be controlled by the experimentor. Examples from simulations of military operations are number of aircraft, weather conditions, or electronic counter-measures.

Factor Levels. The Values assigned to a factor are termed levels. A level may represent a number such as thirty bombers or it may be qualitative. Jamming or not jamming would represent two levels of electronic counter-measures, a qualitative factor.

Treatment. A treatment refers to the level of the factors which describe one simulation run. A convenient notation for

a treatment is ($A_1B_2C_1$). This indicates a simulation run with factor A and C at their lower levels and factor B at the second level.

Response. This is synonymous with simulation results as we have used this term. A treatment response represents the result of a trial of an experiment corresponding to that treatment.

Main effect. A main effect for a factor at two levels is the average difference in response due to the change in levels.

Interaction. An interaction is said to occur when the response to changes in one factor depends on the particular level of another factor. An example might be artillery and spotters. Increasing the number of firing pieces might not increase the damage unless additional spotters were assigned to direct their fire. Similarly increasing the number of spotters might have no appreciable effect by itself.

Sample. A sample consists of a series of plays which differ only as a result of the series of random numbers used. Samples consisting of more than one play are generally used in simulaions whose output is either success or failure rather than a detailed battle history. When a simulation run consists of more than one sample the additional samples represent replications of the same treatment combination. Replication provides the means of estimating experimental error.

The factorial experiment was chosen as the pattern for simulation design because it is generally the most efficient

method of studying the effect on the results, when varying two or more input quantities. [11] An efficient method is one which yields the required information with the minimum effort. In essence the design and analysis of a factorial experiment consists of: (1) conducting trials of the experiment with combinations of several factors at more than one level, (2) observing the response to each treatment and determining estimates of main effects, factor interactions and experimental error, (3) testing hypotheses to determine which of the effects are real and which are attributable to experimental error.

An individual experiment on a completed simulation may be conducted as a complete factorial experiment in which observations are made with every factor at every one of its levels. This usually requires a great many computer runs, however, and some form of fractional factorial experimentation might be employed. The essential point is that by building all of the factors which are pertinent to the study purpose into the model from the very beginning, the capability of efficiently examining any combination of these factors and factor levels will exist in the completed simulation. In this way additional assurance that the analysis will reflect the study purpose will have been gained.

CHAPTER III

PRELIMINARY STUDY

The design of a computer game, as with essentially all operations research studies, must begin with a statement of the problem. In most cases the initial statement of the problem will come from the person or persons requesting the study. The concern here is more with the interpretation of the problem by the simulation designer.

3.1 Formulation of the problem.

The first step toward formulating the problem is to become familiar with the various components of that portion of the real world from which the problem comes. Components in this context may include hardware such as weapon systems, operating doctrine, operational environment and a myriad of other items which make up the particular aspect of the real world under consideration. It is suggested that a block flow diagram showing the functional position of each component in the system be constructed as part of this initial research. A flow diagram is also useful for showing interactions among components.

The determination of the scope of the proposed design should be concurrent with the preceding step. As the analyst familiarizes himself with the system components, the size and complexity of the problem area will become more apparent. Conversely, unless he has some feel for the range of activity included in the problem it is impossible to know what the components of the system are.

If the design is to be built around particular questions as related to the objectives of the system, these questions must be clear from this point on. It is therefore prudent to compare the interpretation of objectives with stated objectives. It is possible that on the basis of the above initial research, the original problem may be modified or expanded, or another research method chosen.

3.2 Applicability of computer simulation to the problem.

Once the scope of the problem is known and the designer has gained some understanding of the system, it must be decided whether or not computer simulation techniques fit the situation. This is a step which is perhaps too often omitted in simulation design. The following statement by C. J. Thomas concerning unfortunate practices in military gaming is equally applicable to computer simulation.

> ...there is an unfortunately common tendency to think of "military gaming" as being not only universally applicable but also mechanically applicable. The failure to match technique with problem is often expensive and almost invariably produces disappointment. [13]

Consideration of theoretical model. It is generally considered that simulation should be used only if it is impossible to solve the problem using a simple theoretical model. [14,15,16] In applying this generality it is important that we do not lose sight of the given problem in trying to fit the system to an analytic model. Any situation can be treated analytically if it is simplified enough. However oversimplification of the model and the subsequent analysis may invalidate the results of the study. Consider the helicopter search situation treated by AHS-1. The system itself

is not overly complicated, and analytic models of a search helicopter versus an evading submarine conflict have been built. One of the helicopter search plans currently operational with the U. S. Navy is based on a game theory solution of this conflict situation. The model used in this solution includes the assumption that submarine maximum speed is known, and that helicopter sonar ranges and dip cycle time are known, and the same for all helicopters. The simulation study, AHS-1, was motivated by the fact that these factors among others are subject to uncertainty. The author wanted to know the relative effectiveness of certain search plans, including the so called optimum plan found analytically, in the face of this uncertainty.

Expected number of plays. A second consideration in deciding when to resort to computer simulation, is the number of expected repetitions of the experiment. The validity of the Monte Carlo technique is dependent on repeated trials. It is not unheard of to make ten or more consecutive sevens with two fair dice. Inferring that this is routine on the strength of observing the phenomenon once, however, could be financially disastrous. Similarly, confidence in simulation results is related directly to the number of replications.

The number of input parameters to be examined and the range of interesting values of these parameters also affects the number of required repetitions. The effect on experimental results of varying parameter values is usually of more interest than the particular outcome corresponding to any one set of input values. In addition certain parameters may exhibit

interaction. That is the effect of changing the input value of two or more factors simultaneously is considerably different from that of varying these same parameters individually.

It is apparent that if several parameters are to be examined at more than one level, each run consisting of many replications, the number of simulation runs can become very large. Even with a game of relatively limited scope such as AHS-1, several months might be required to examine all input parameters at only a few levels by manual methods. On the other hand, if only a few plays of a game are contemplated, computer simulation is probably not the most efficient method. A completed computer program represents a sizeable investment and the cost in time and money of running this program on a computer one time represents a small fraction of this investment.

Player training. The potential value of training derived from player participation in the game must also be considered. If the primary purpose is clearly player training then a manual game is indicated. Whether the purpose is educational or analytic is not always tacitly obvious, however, and a decision to satisfy one or the other requirements will have to be made.

Assuming that computer simulation is theoretically applicable the analyst must next ask if it is physically practicable. A study which is worth doing is worth doing correctly. If the limitations of time and available phisical faci-

lities preclude an adequate treatment of the problem by simulation methods the person requesting the study must be apprised of this.

3.3 Research into existing simulations.

Having decided to use simulation methods, the designer might next ask if there is an existing model which is suited to this particular situation. However, in many respects trying to fit the problem to an existing model is contradictory to the stated objective of tailoring the simulation to the particular study purpose. It would be preferable to restate the question more generally as; what is the existing level of simulation knowledge which may be related to this system? Although it might be unwise to modify the particular situation to fit an existing model, it is equally foolish not to profit from the effort of others whenever possible.

To illustrate this point we refer to the scheme by which sonar detection ranges are computed in AHS-1. The particular objective required some more realistic means of determining detection ranges than the usual procedure of using a deterministic assured sonar range. The scheme selected was designed by R. L. Klinkner as a general method, applicable to most existing Navy sonars.[17] Furthermore an algorithm for computing the needed sonar ranges by this method had been written and coded in the computer language previously selected for AHS-1. The availability of this computer coding made it possible to design a simulation which fit the requirements of the problem within the time allotted. This might well have been impossible otherwise.

CHAPTER IV

DESIGN OF THE MODEL

The model of interest in this chapter is a stochastic mathematical model. Recall that a mathematical model was defined as a mathematical representation of some aspect of the real world. This model can usually be subdivided into models of each of the individual components or activities which characterize the system of interest. It is partly because of this subdivision that the construction of a block flow diagram was suggested as part of the formulation of the problem. Such a flow chart is often a useful aid to recognition of logical functional subdivisions.

4.1 Functional components of the model.

There are several reasons for subdividing the model into functional components, the degree to which this is done being closely related to the scope of the system under study. Any system, even one of limited complexity such as the helicopter-submarine conflict of our example, is much simpler to visualize if it is broken down into functional components or activities.

A second argument for subdividing the model along functional lines is that for more complex problems several persons may be simultaneously engaged in modeling the system. Consider a simulation of a large area anti-submarine operation. Helicopter operations might constitute one logical subset to be allotted to one individual. One person could develop

this part of the simulation, code it and complete the program check-out independently of the main study group. The example, AHS-1, could with only minor modification represent such a subsection of the larger simulation.

Dividing the model into such subsets also facilitates parameter validation and program check-out. This applies equally whether the simulation is being built by one person or several.

The preceding model subdivisions are primarily for mechanical or operational convenience of the designer. They are aids to organizing the existing knowledge about the system so that the model will become physically manageable. Models may also be subdivided into conceptual, or structural, components.

4.2 Structural model components.

The structural, or conceptual, components of a model are:

(1) Assumptions

(2) Input parameters

(3) Input variables

(4) Algorithms

(5) Measures of effectiveness

It will be simpler to define the above terms and explain why they are considered to be useful to the model designer if the following two definitions are agreed upon. These definitions are from Van Nostrand's Scientific Encyclopedia. [18]

> Parameter. An arbitrary constant, as distinguished from a fixed or absolute constant. Any desired numerical value can be given to a parameter.
>
> Variable. A quantity as distinguished from a constant to which any number of values in a given set may be assigned. If the set comprises a domain(a,b), then the variable

is only defined over this interval and all values outside of the interval are ignored.

These definitions will apply whenever parameter or variable is used in the succeeding discussion.

Assumptions. Any conclusions arrived at on the basis of simulation results must be considered invalid unless they were made with a complete knowledge of the major model assumptions. The model assumptions reflect more than any other factor the designers understanding of the system to be modeled and his interpretation of the objective of the simulation. The assumptions reflect what is known precisely about the real world as well as what is uncertain and the degree of uncertainty. To illustrate, the following are model assumptions from AHS-1.

The ocean is represented as a rectangular coordinate system in the simulation. This is based on the assumption that the area encompassing a helicopter search is small enough that the earth's curvature need not be considered. Helicopters, which are characterized as points from which associated sonar ranges are measured, are assumed to move about this square grid in discrete jumps. These assumptions reflect certain well defined facts about helicopter sonar search operations.

It is further assumed that the above sonar ranges vary among helicopters according to some probability distribution, and that this distribution is a function of sonar figure of merit. It is known that sound propagation conditions, at any one point in the ocean, are a function of a time correlated random variable. [17,19] This time dependence of sonar

is only defined over this interval and all values outside of the interval are ignored.

These definitions will apply whenever parameter or variable is used in the succeeding discussion.

Assumptions. Any conclusions arrived at on the basis of simulation results must be considered invalid unless they were made with a complete knowledge of the major model assumptions. The model assumptions reflect more than any other factor the designers understanding of the system to be modeled and his interpretation of the objective of the simulation. The assumptions reflect what is known precisely about the real world as well as what is uncertain and the degree of uncertainty. To illustrate, the following are model assumptions from AHS-1.

The ocean is represented as a rectangular coordinate system in the simulation. This is based on the assumption that the area encompassing a helicopter search is small enough that the earth's curvature need not be considered. Helicopters, which are characterized as points from which associated sonar ranges are measured, are assumed to move about this square grid in discrete jumps. These assumptions reflect certain well defined facts about helicopter sonar search operations.

It is further assumed that the above sonar ranges vary among helicopters according to some probability distribution, and that this distribution is a function of sonar figure of merit. It is known that sound propagation conditions, at any one point in the ocean, are a function of a time correlated random variable. [17,19] This time dependence of sonar

ranges has not been included in the model for two reasons. First, a helicopter search occupies a relatively short time span and the variation due to the above random variable is expected to be small in the time interval of interest. Secondly, sonar ranges can be expected to decrease or increase with the same relative frequency during a search. If for some reason sonar ranges were biased so that either a decrease or an increase with time could be expected this assumption might not be valid.

The assumptions concerning sonar range all have a bearing on uncertainties in nature and the effect of those uncertainties on the simulation purpose. All of the preceding assumptions reflect the model builder's understanding of basic functional relationships of the system components as they are related to the simulation purpose.

Input parameters. These are constants which may be assigned any arbitrary value by the simulation user. Our interest in input parameters is generally in the way changes in parameter values affect the outcome of the operation simulated. The factor which characterizes these quantities as parameters rather than variables is the degree of human control in the real world situation being simulated. For example, the particular search plan designated by a helicopter flight leader is entirely within the control of the flight leader in question. In assigning values to this particular input parameter the analyst is concerned with finding criteria by which a certain plan should be selected, given a specific state of nature. Note that input parameters need not be

numerical quantities nor will they necessarily be the actual inputs to the program. Search plan is an input parameter and its value might be Plan Lisa or Spiral Search Three. This quantity will give rise to input quantities which do have numerical values, however. In this example bearings and distances of the search legs are such numerical valued quantities.

Input variables. These are input quantities over which we cannot expect to have control in actual practice. For example submarine speed, sonar conditions or time late at a submarine datum. Input variables are characteristic of the system and the range of values assigned to these quantities is essentially determined by nature.

As noted above for input parameters, input variables as initially defined by the model designer may not be the actual program inputs. In most cases these will be determined in conjunction with the detailed investigation and data collection intrinsic to the actual model construction.

To tie the above definitions together, consider the functional relation from AHS-1:

$$R_o = F(S_s, T_1; P_A, H_N \ldots)$$

where R_o = Output (Number of detections during a run)

S_s = Submarine Speed
T_1 = Time late of helicopter at datum
} Input variables

P_A = Search Plan A
H_N = Number of helicopters
} Input parameters

F is the rule, determined by the model assumptions and simulation logic, which determines the dependence of R_o on the input variables and parameters. Submarine speed and time late are independent variables which are allowed to vary over selected portions of their assumed ranges. The input parameters may be assigned any arbitrary set of values. Whereas the range of input variables depends on what the analyst has assumed to be the possible states of nature, input parameters reflect particular questions about the system. The optimum values of these parameters corresponding to specific states of nature, i.e., specific values or ranges of values of the input variables, represent part of what we hope to learn from the simulation.

Input parameters and variables may change roles according to the specific circumstances of application. For instance, it may be possible to control time late at datum to some extent but at the expense of the number of helicopters which can be sent to the datum. It might happen that two helicopters are airborne when a submarine datum is established. The decision must be made to launch more helicopters and accept a greater time late, or proceed immediately to datum with the available aircraft. Here time late assumes the role of a parameter and number of helicopters is essentially a variable.

Algorithm. "An algorithm is a stated rule or procedure that may be followed to arrive at some specified goal."[20] An algorithm is characterized by the fact that it may be

mechanically applied. The procedure for generating random numbers employed in AHS-1 is an algorithm.

The model designer must consider every possible question which might arise and preprogram an answer to each question. It is not possible to defer a decision until such time during the play of the game that the need for making the decision may arise. Therefore, in order to build a model which can be incorporated in a computer program it must be possible to reduce the essence of the real world situation to a collection of algorithms. By means of this set of algorithms the program operates on input variables and parameters in a way determined by the assumptions.

Measures of effectiveness. Saaty defines measure of effectiveness as "a criterion which measures the extent of success of a solution, as related to the objectives, when applied to a problem arising in an operation." [21] The principle measure of effectiveness incorporated into AHS-1 is number of submarine detections. In this example the objective, as used in the above definition, is to evaluate the comparative effectiveness of various search plans in similar tactical and physical environments. One of the problems in this example is that the number of available helicopters is generally limited.

4.3 Determining measures of effectiveness.

Military simulation studies fall into two broad categories according to purpose; testing plans and evaluation of weapon systems. Regardless of where the simulation fits in this general classification, however, if decisions are to be

influenced by the simulation results some measure of the success or failure of the operation in a given instance is required. Measures of effectiveness may be comparatively obvious, or quite subtle. In the case of AHS-1, the number of submarine detections is a straightforward measure of the comparative effectiveness of two search plans under similar circumstances.

Consistent well defined measures of effectiveness are essential to good simulation design. To a considerable extent the measure of whether an included parameter or variable is essential to the simulation purpose is the sensitivity of the measures of effectiveness to changes in this input. Consequently, the measures of effectiveness will represent important criteria by which the analyst must decide whether a factor is to be included or omitted from the simulation.

Inconsistent measures of effectiveness indicate incompatability of the simulation objectives as well as the lack of a clear interpretation of the objectives of the system being simulated. In essence the measures of effectiveness are the statements of the objectives of the system being studied. The selected measures of effectiveness are indicative of the designers understanding of the system objectives just as model assumptions reflect his understanding of basic functional aspects of the system.

Output data. As far as is possible the output of the simulation should be determined in conjunction with selecting measures of effectiveness. All of the measures of effectiveness will be output quantities and these may also give rise

to associated numbers which will be of interest to the analyst. In addition, certain control variables should be included in the output. These controls should be quantities whose values or distributions are well enough known that they can be used as rough checks on the logic and numerical inputs to the simulation.

4.4 Selecting inputs

Everything that has been accomplished up to this time has been preliminary to the actual design of the model. During the formulation of the problem the scope of the simulation was determined. The scope in turn determined the acceptable level of aggregation consistent with the stated problem. Having chosen measures of effectiveness by which the simulation results, and to some extent the simulation itself, will be evaluated; the designer is prepared to select the inputs and incorporate these factors along with certain assumptions and rules into a representation of the system.

Inputs were classified as either variables or parameters on the basis of the extent to which these quantities can be controlled in the real world. This classification will be useful in determining the ranges of input variables and parameters and the distinction should be kept in mind during the following discussion. In determining which inputs to include in the model, however, it is more suitable to our purpose to combine inputs and variables under the single title of factors.

D. R. Cox lists four items to be considered in designing a factorial experiment.[12] These are, the factors to

include in the design, the levels of each factor, the number of observations to be made, and measures to reduce the effect of uncontrolled variation. The last two of these are more closely related to simulation testing and analysis of the simulation results. The factors to be included and the levels of each factor determine the form of the model itself and will be considered at this time.

It should be noted that at this point the simulation designer is concerned with a general experimental design. One particular experiment might involve only two or three factors, each at one or many levels. However, the simulation objective often involves several questions about the system, and the analyst must design the simulation to encompass a class of experiments. Consequently, all of the factors necessary to conduct any experiment of this class must be built into the model.

Choosing factors. Whether a particular factor should be included in the simulation is primarily a matter of experience and judgment. Once a working version of the simulation has been completed, statistical methods can be employed to test the sensitivity of the measures of effectiveness to the selected inputs. In addition a certain amount of parameter testing can be accomplished by hand or desk calculations as the model is being built. For the most part, however, factors must be selected on the basis of experience, that of both designer and user, conditioned by what was learned during the preliminary investigation of the problem.

An example from AHS-1 may lend more meaning to this assertion. Probability of gaining a non-submarine contact and maximum time to evaluate such contacts are included as input parameters. Actually these represent any delay experienced by a helicopter while conducting a sonar dip; non-submarine contacts are one of many reasons for such delays. That delays occur is well known to those who have participated in helicopter search operations. However, the existence or significance of variation in dip times would not necessarily be apparent to an analyst who had not participated in anti-submarine operations, solely from a review of the literature.

This introduces another question to be considered by the model designer. To what extent may too close association with the day to day operation of the system limit the perspective of the simulation customer? For example, a catapult officer may be a valuable source of information concerning aircraft carrier air operations. However, this officer is involved with many factors which are essential to air operations but may not be pertinent to the purpose or level of aggregation of a simulation of air operations. Gaining the maximum information from this officer, without cluttering the simulation with unnecessary detail also based on the man's experience is again a matter of judgment.

The designer must also appreciate the role of the decision maker when considering inputs. It is the user who must take the final responsibility for decisions based on the simulation results, and hence these decisions will be influenced to

a large extent by his confidence in the simulation. In the above example if exclusion of catapult pressure from the simulation will result in a complete lack of faith in the simulation by the decision maker, then catapult pressure may be a valid input even if the simulation results are insensitive to this input. It may be that only by including this factor and conducting a more detailed analysis of the sensitivity of response to it can the question be resolved to either the designer's or user's satisfaction.

Factor levels. In selecting factor levels, just as with the factors themselves, the designer will usually want to include a certain amount of generality in the model. The factor levels chosen at this time will in some cases be the particular levels used for the analysis later on. In most cases, however, the designer will be interested in a range from which specific levels may be chosen at the appropriate time. In this phase of simulation design, planning the experiment around specific questions can be particularly productive. If only certain selected ranges of factor levels have a bearing on the study objectives, time spent determining the entire range of possible values might be essentially wasted. Restricting the levels to interesting ranges can also result in many efficiencies in the subsequent computer program. As the range of factor levels is increased so is the computer storage space necessary to contain the associated numerical values.

Determining factor levels will also give rise to the need for determining certain distributions of associated

input variables. It is in this step that the data collection role of statistical analysis is of particular interest to the simulation designer.

Input parameters are directly related to the particular questions posed by the person or activity requesting the simulation study. Consequently the customer is often the best source of the range of parameter values. Input variables, however, are more closely associated with the possible states of nature. The designer is free to include or reject a specific variable according to its pertinence to the simulation purpose. Once he has included a particular variable in the model, however, the analyst must determine the range of values of the variable according to what actually exists in the real world. This determination might involve a comparatively simple library search or an extensive analysis. Determination of factor levels may at times give rise to analytic or simulation studies which are more extensive than the original simulation.

It might happen that the required information cannot be obtained in the time allowed or at a cost consistent with the designer's operating budget. Perhaps the information is not available at any cost, or only a very restricted segment of the true range of the variable can be determined with any degree of certainty. The designer may leave this variable out or he may resort to an educated guess. What is done is a matter of judgment and depends on the particular situation. It is essential, however, that the simulation reflect the

nature and degree of uncertainty relative to this input. As with any analysis the results have validity only with respect to the assumptions and input data from which they come.

4.5 Combining factors and assumptions in the model.

As an illustration of how the assumptions, factors and factor levels, and algorithms specify the model, we will consider sonar detection ranges in AHS-1. From the preliminary study it was determined that the simulation purpose dictated that sonar detection range was an essential element of the helicopter model. This quantity is essentially dependent on nature and was therefore defined as an input variable.

Further investigation revealed that detection ranges depend on certain variables such as figure of merit, water conditions and sea state. These were tentatively assigned as factors and the process of determining general factor levels was commenced. As noted above, assigning factor levels is closely related to determining the range of input variables, and data collection.

The first step in assigning general factor levels was to determine the functional relationship between the original input variables, sonar detection range, and the tentatively selected factors. Investigation in this area revealed the stochastic nature of figure of merit and resulted in a modification of the selected inputs. That is the distribution parameters, average figure of merit and variance, replaced figure of merit as inputs to the simulation. This of course involved certain assumptions which have already been mentioned.

On the basis of the above research and assumptions the valid range of factor levels was determined. It was then only left to discover algorithms, by which the specific values of the original input variable could be computed, and combine these with other algorithms which would move the helicopter model over the playing area. It was of course necessary that these algorithms not only be mathematically correct but that they be applicable over the range of both factors and the original input variable. The end product of this process was a point which could be moved about a square grid, and an associated detection range. This is one model of a helicopter.

CHAPTER V

IMPLEMENTING THE MODEL

5.1 Adding logic to the model.

The model is the heart of the simulation, but to implement the model it is necessary to move the simulated system components through time as well as space. The simulation logic is the means by which the model, a static representation of the system components, is transformed into a dynamic time ordered simulation of the system. In practice the design of model and logic may be concurrent, but conceptually there exists a distinct interface between them.

To illustrate this, we refer to the helicopters in AHS-1. As stated earlier, a helicopter is represented by a point on a rectangular grid together with an associated sonar detection range. These, combined with a suitable algorithm for changing the grid position of this point, complete the model of a helicopter. The implementation of this model in a simulation of a helicopter search operation requires some procedure for causing certain actions to take place at specified times. Each helicopter must move to successive dip stations at prescribed times. Having arrived at a dip station, a search for the submarine and a determination of the outcome of that search must be made. To complicate this picture, other helicopters are simultaneously acting and interacting with the target submarine.

The determination of which actions are to take place and their relative order of occurrence could be accomplished manually, using the above model. Similarly, logical structures other than the one used in AHS-1 could be employed to implement this same model in a computer simulation.

The two most frequently encountered logical structures are the event store, employed in AHS-1, and the time-step. Both of these structures are essentially techniques for approximating time, which is continuous in the real world, by a series of discrete steps. In an actual operation certain events take place either simultaneously or in an overlapping time space, but a computer can only perform one operation at a time. Consequently when an action occurs in the simulation, time must be arrested for all units not participating in that action. Some adjustment must then be made for interacting events which occur simultaneously. This necessitates some predetermined logical order in which actions and participating units are to be considered during the play.

Time-step. In a time-step simulation the play takes place as a series of discrete time increments. The length of each interval is called the time step and may be thought of as representing the length of one game move.

During each time step the positions and capabilities of all units are noted and a list of possible actions is consulted. For each entry in the list of possible happenings, all units which could be affected by this action are then considered in turn. With respect to each of these units,

calculations are performed to see if the action in question did in fact occur and to what extent the unit was affected. After every action in the list has been considered, the positions and capabilities of all participants are reassessed and the play moves into the next time interval. The end of a play may be signalled by the occurrence of some predetermined action or at the completion of a specified number of time steps.

It is characteristic of this method that all events which occur within the same time step are considered to have taken place either simultaneously or in the order in which they appear in the list of possible actions. For example, two possible events might be weapon detonations and missile launches. If missile launches are considered ahead of detonations, then it would be possible for a missile to launch when in fact the missile site had been destroyed by a weapon detonation prior to scheduled launch time. The probability of this happening can be decreased by decreasing the length of the time step, but at the cost of increased computer running time.

Event Store. An event store simulation is played as a sequence of events rather than discrete time intervals. In contrast with the time-step method, these events are not considered until they are scheduled to occur. The scheduling of future events is accomplished by entering elements, commonly referred to as event words, in a table called the event store. Each event word consists of the type of event, the

time the event is to take place and any other information which is required for the execution of the event. In the sample program, AHS-1, the identity of the unit involved in the event is part of this word. In many simulations, events once entered in the list may be cancelled as a result of subsequent action, necessitating the inclusion of a validation signal in each event word. Events which have been invalidated may then be discarded when they reach the top of the event store.

It is characteristic of the event store logic that sections of the program representing events are essentially independent of each other. Interactions among these separate components are caused to take place by two events usually referred to as Take Next Event (TNE) and Store New Event (SNE). Store New Event enters event words representing future events, in the event store in chronological order, and Take Next Event causes events to be executed at the appropriate time.

At the beginning of a play certain events which can be predicted from the program logic and the input data are entered in the event store. Control of the program then passes to TNE which notes the event type of the earliest valid event word and calls the appropriate event subroutine into action. This event subroutine carries out its preassigned function according to the information contained in the event word and the input data. As a result of this action other events may arise, in which case SNE is called upon to store the necessary event word. At the completion of each event, control is returned to TNE and the cycle repeated. The play terminates

when some particular action occurs, such as a submarine detection in AHS-1, or when there are no more events in the event store to be processed.

Time-step versus Event store. Efficiency of both programming and computer operation are usually prime considerations in choosing the logic for a simulation. The event store method is somewhat more compatible with the functional subdivisions of the model previously mentioned. Each event subroutine usually corresponds to some particular function of a system component. On the other hand, the programmer may have difficulty understanding the chronological sequence of events at first. [22]

Since both structures are methods of approximating real continuous time by discrete steps, the number and hence the length of each step is closely correlated with machine running time. With the time-step procedure one time step may be considered as a single approximating step. Every possible action and every unit is considered once each time increment, so as the number of time steps is increased, machine running time becomes greater. Decreasing the number of operations by lengthening the time step results in greater time aggregation which is usually undesirable. In general the time-step method results in greater computer running time when the exact order of occurrence of closely spaced events is a critical consideration.

In the event store method each event may be considered as one approximation. With this structure, actions begin in

the proper time sequence but in the machine only one event can occur at any one time. Consequently interactions among events which occur in overlapping time intervals may be lost. An example of this is the relation between submarine maneuvers and submarine detection events in AHS-1. A detection event consists of one or more sonar searches. If the submarine is due to maneuver while a detection event is in progress, then the correct submarine position will not be available from the time the maneuver is scheduled until the completion of the event being processed. The effect of this could be minimized by defining each individual search as one event. This would essentially increase the number of events and would result in increased machine running time. The procedure that is used in this case is to interrupt any detection event between searches if a target maneuver is to occur. This is determined by asking if the submarine maneuvered at the end of each search. The submarine is then maneuvered and the detection event resumed at the time it was interrupted.

The above is an example of superimposing a time-step on one portion of the event store logic. Similarly an event store may be incorporated into a basic time-step structure. This technique is employed in STAGE, a global atomic exchange model built by Technical Operations Incorporated under contract to the U. S. Air Force. The STAGE simulator, which is played in fifteen minute time intervals, uses the output of a preprocessor for its input. Part of the STAGE preprocessor output is a list of events which have positive probability of

occuring during each time interval. This list is read from magnetic tape into the main simulation program as it is needed and essentially represents an external event store. The simulation itself then "Steps through each time period, considering the various events scheduled to occur in each time period and determining which events actually occur." [23]

Flow charts of the logic. Block flow diagrams can be very useful in organizing the logic as they were in delineating the functional subdivisions of the model. Flow charts also provide a convenient vehicle for communication between simulation designer and programmer.

Flow charts at this stage of the design will usually be constructed on at least two levels of detail. General flow charts consisting of boxes enclosing plain English statements are one means of describing the model and the logical way in which model components interact. General flow charts, along with the documentation of the structural model components may completely specify the model and logic. However, more detailed flow diagrams must usually be constructed before a professional programmer can reduce the model and logic to a useable computer program.

5.2 Computer languages

Having reduced the model and logic to detailed flow charts the designer is ready to turn his creation over to a programmer, or as is sometimes the case to write the computer program himself. A computer program is essentially a list of instructions written in a format, or language, which is useable by the computer. The program is the vehicle by which a

complicated mathematical model is reduced to the very limited set of operations which can be performed by a computer.

A complete hierarchy of computer languages is available and the selection of one of these depends on several factors. Some of these considerations are: (1) simulation size,(2) expected number of runs of the completed simulation,(3) programming facilities, (4) available time for programming and program testing, and (5) the available computer facilities.

Computer languages may be classified according to relative programming ease. In general the simpler the programming the more inefficient the program is in its utilization of operating time and memory space. One possible classification is as follows:

1. Binary code
2. Assembly language
3. Compiler language
4. Simulation oriented language

Binary code. This is the only language the computer actually "understands" and all other language systems must ultimately reduce the programmers instructions to the binary number system. Writing binary instructions is extremely time consuming since complicated operations must be reduced by the programmer to a sequence of basic operations which the computer itself can accomplish. These basic operations consist of storing and transferring information, addition, subtraction, multiplication and division.

Assembly or symbolic languages. These languages, such as IBM FAP and the symbolic language used with the CDC 1604 employ what is called an assembly program. The programmer

writes assembly language instructions which are then translated into machine instructions by the assembly program. The programmer might write a statement such as FAD (R), which is symbolic for Floating Point Add (Location R). This statement instructs the computer to note the contents of a location previously designated as R and add it to what ever has been placed in a section of the machine called the A register. This sum will then be left in the A register and may be stored in another location by additional instructions.

This is a much simpler instruction to write than the corresponding binary configuration. In addition, the number of the location in which R is stored need not be known to the programmer, as the assembler will keep track of R once it has been defined. When binary machine language is used the programmer must keep track of every location and its contents by number.

Compiler languages. The use of languages such as FORTRAN or ALGOL further simplify the programmer's task. In general, however, the step from assembly to compiler language involves a considerable loss in storage space and flexibility, and an increase in operating time. The FORTRAN program is written as a sequence of arithmetic statements such as R = R+A. This means take the contents of the location assigned to R, add this to the location assigned to A and place the answer back in R. Here the format is again floating point but this is signaled to the compiler by the particular letters used.

The FORTRAN program, consisting of statements such as the one above and many other instructions which are a great

deal more complicated, must first be compiled. This consists of a compiler program translating the FORTRAN statements into an assembly language program. This assembly language program is then converted to machine instructions. This generally involves inefficiencies both in the utilization of time and storage space. The compiler, lacking the ability to reason, is not as sophisticated as a human programmer and tends to waste instructions to some extent.

Compiler languages do have some very important advantages in addition to ease of programming. One of these is the large library of function sub-routines and programs which is available. For instance the simple statement R = SINF(THETA) can be used to compute the sine of an angle THETA and store this value in the location R. To accomplish this using an assembly language, the programmer would have to write the instructions for computing the Taylor's expansion of sine THETA, add the terms out to some desired accuracy and place the sum in location R.

Simulation oriented languages. Languages designed primarily for simulation, such as SIMSCRIPT or MILITRAN are currently being developed. [24,25] These languages, or programming systems, are designed to further simplify programming. In general, the program is written as a series of functions comparable to the event subroutines in AHS-1, rather than arithmetic statements. The source program consists of phrases similar to those commonly used in general flow diagrams. These statements are then translated into subroutines written in a core language such as FORTRAN.

Choosing a computer language. Probably the first consideration in selecting a language is size of the resulting program. If the problem will not fit into an available machine, running time and programming considerations become academic. If the simulation is large, however, the program size should be considered in conjunction with available data processing facilities. In many cases the effective machine size can be expanded far above actual size through the use of satellite equipment such as magnetic tape drives. Parts of the program and input data may be stored on magnetic tape external to the machine. The taped information is then read into the computer only when it is needed.

If space requirements indicate that a compiler language is feasible then the balance between machine operating cost and programming expense is very important. If a simulation is to be run only a few times, a compiler language is probably indicated because of the lower programming cost. Of course if programming in assembly or machine language would delay the completion until after the information was no longer of interest, a compiler program might be the only alternative no matter how many simulation runs were anticipated.

The availability of existing subroutines which may be useable in the program should also be given careful consideration. In selecting a language for AHS-1, the programmer was obliged to choose between two compiler languages, FORTRAN and NELIAC. This restriction arose because the program was intended as an example and these two languages are the more commonly used within the United States Navy. The final

decision to use FORTRAN was greatly influenced by the availability of a sonar range prediction program written in FORTRAN.

One of the most important drawbacks to FORTRAN for simulation studies is the inability to pack more than one quantity into each word. For example the event store table used in AHS-1 consists of three arrays. In assembly language these could be combined into one table by allotting only part of a word to each of the three items of information. It is possible to write packing and unpacking routines in assembly language and combine them with a FORTRAN program. Such routines take time to program and it might be almost as convenient to write the entire code in an assembly language.

The various dialects of ALGOL, such as NELIAC or JOVIAL, do retain the packing feature of an assembly language, but at the cost of an increase in computer operating time.

5.3 Program check-out

Regardless of the language in which the computer program is written, it can be expected to contain some errors. Careful planning and attention to detail will go a long way toward minimizing these errors, but a computer code is an extremely complicated structure, and the specifications for computer programs are very exacting. The omission of one instruction or even one decimal point may result in the entire program being rejected by the computer.

It is probably easier to test the program if it has been constructed in independent sections, and it was partly toward this end that the model was divided into functional components. Each of these sections provides a particularly convenient unit when testing the program because of the relation between input

data and individual functional components. If the coding is tested in blocks, input data which would normally be provided by other sections of the program is usually introduced by a check-out control program. This may consist of a few cards containing the necessary data or might incorporate previously tested subroutines. In this way, data which is input to the subsection being tested can be precisely controlled. Localizing an error in a program section which uses self generated input data is complicated by the fact that both the program logic and input data are possible sources of the error.

Although coding the program in subsections is probably the most significant single aid to program testing, it is not a complete panacea. Though each section performs properly over its designed range of inputs, this does not insure that all of these components will interact with one another as the designer intended. One method of testing the completed program is to compare a limited number of runs with hand calculated results. This will generally involve checking the program at certain critical points. The procedure used in testing AHS-1 was to have the values of key variables printed at certain points throughout the program. In this way the output from one operation and the input to the next link in the chain were checked simultaneously. These were then compared with hand calculations.

The above procedure is a tedious and time consuming task for even a relatively uncomplicated program, but it may be the only way the simulation designer can be reasonably confident of the correctness of the program. Even when such a

testing procedure has been completed there is generally no way of knowing with one-hundred percent confidence that the program is free of errors. Such assurance would ordinarily involve testing every possible combination of input values, a virtually impossible task.

5.4 Auxiliary program components.

The questions of what to include as input and output data confronted the simulation designer throughout the model design phase. The question of how to get this information in and out of the computer must also be answered. With the capacity to process large amounts of data inherent in a digital computer, data processing is not a trivial consideration.

The input data format will often be influenced to a considerable extent by the size of the program. Space limitations may necessitate reading data in only as it can be used. If space is not a problem, time may be saved by reading as much data at one time as is practicable and storing it internally.

Space and time considerations will affect the particular form of input data as well. In a small simulation, data may be introduced in a form which is more convenient to the user and any necessary conversions accomplished internally. Large simulations might require considerable pre-processing of data to conserve room for calculations which can be conducted only in conjunction with the operating section of the program.

Errors in the input data are also an important consideration. Data input to AHS-1 is printed out as a check on what values were actually used by the simulation, but an important

section of the program was omitted because of time limitations. This omission is a routine for checking the input quantities to see that they fall within acceptable limits. This would not eliminate mistakes entirely but could reduce lost computing time due to misplaced decimal points and other gross errors in the input data.

The output section of the program will be influenced to a considerable extent by the simulation objectives. In addition to selecting outputs which will yield the maximum information, the designer should specify an output format which is compatible with the particular analysis techniques to be employed. For example when space permits, computation of sample means and variances might save the analyst a great deal of time. Similarly, percentages or ratios of numbers are often more meaningful than the numbers themselves.

5.5 Testing the simulation.

In designing the model, factors were included because they were known, or in some cases only suspected, to affect the system in a way which was important to the study purpose. Before the simulation is ready for use as an analytic tool, the designer should determine both quantitatively and qualitatively how these factors affect simulation results. This will require that some testing program be conducted and representative levels of all or part of the factors examined.

Requirements of a testing program. The testing program may include a comparison of simulation results with observations of the system itself, or with experimental results obtained by other analytic methods. It should include some analysis of the sensitivity of measures of effectiveness to

changes in factor levels. Whatever the analytic techniques employed, the following appear to be fairly general objectives of the tests.

(1) Verify that simulation response to well known factors is in general agreement with observation.

(2) Determine the sensitivity of response to all factors.

(3) Determine the number of replications necessary to attain certain levels of statistical validity.

(4) Determine which factors interact and the extent of such interaction.

Conducting the testing program. The number of runs necessary to attain the desired accuracy should be one of the first considerations during simulation testing. In testing the simulation, as well as in conducting subsequent experiments, the analyst is usually guided by two major considerations; how to gain as much useful information as possible from the simulation, and still minimize the expenditure of time and resources. Testing should also provide some general guidelines as to the number of replications necessary to achieve certain levels of confidence in the simulation results. These may then be applied during both employment and testing of the simulation. Testing the simulation using too few runs may essentially invalidate the results of the tests.

When comparing simulation results with observations of the system, it is well to consider the source of such observations. In analyzing military operations the results of peace time exercises may be the nearest the analyst can attain to an actual observation. Field maneuvers and fleet exercises may at times approach reality, but the fact remains that they are another

form of simulation. As with real battles, an exercise may be conducted only a few times. Consequently the analyst must consider the dangers associated with drawing inferences from small samples. The generality of the model must also be considered when comparing simulation response with observations from the real world. Agreement between simulation results and one particular observation may reflect that the designer has only simulated that particular observation.

During the initial testing it may be noted that certain combinations of factor levels tend to saturate the simulation output. That is, some factors because of their greater effect on the simulation response, may tend to completely determine the results when assigned values near the extremes of their ranges. Knowledge of these saturation levels should be useful when conducting sensitivity studies on the remaining input parameters and variables.

Since the simulation was designed as a factorial experiment from the start it is expected that this method of analysis will play an important part in the sensitivity testing. Factorial experiments are particularly adaptable to sensitivity analysis as well as to the determination of factor interactions. [10,26]

R. J. Matteis and W. C. Sunler have described a testing program employed in sensitivity testing of the Carmonette Model. [26] The testing program was conducted in two phases: The first phase consisted of examining the more important factors and generating some estimate of effects and variances. In this phase, factors which exhibited saturation effects were treated

individually. The remaining factors were then examined in a complete factorial design. In the second phase, knowledge gained during the initial testing was used in designing a fractional factorial experiment to test the overall model for interactions.

5.6 Documentation.

Documentation can very easily be the defining line between a partially completed computer simulation and a useful analytic tool. The importance of assumptions has been emphasized throughout the preceding discussion of model design. Unless these assumptions are written down, however, they may not be considered during the analysis. This is further complicated by changes in personnel or by physical separation of designer and ultimate user.

Properly documenting program logic and computer code is no less important than completely describing the model. The real world is dynamic and the simulation must often reflect change. Attempting to modify a program without complete flow charts and a dictionary of variable names, as well as the major model assumptions, can be a formidable task. In some cases it may be simpler to rewrite the coding than to modify an existing program.

Standards of format and adequate content vary with each organization engaged in simulation design, and standardization among simulation groups is virtually non-existant. The description of AHS-1 reflects the author's own idea of adequate documentation, with one exception; the simulation has not been thoroughly tested and no mention is made of tests results. In summary, the following list is considered to contain the minimum requirements for documentation of a computer simulation.

(1) A statement of all major assumptions pertaining to the model with appropriate references, and a description of the model.

(2) A description of the logic. This should include general flow charts.

(3) Complete rules for input data preparation, and the allowable range of input values.

(4) Statement of measures of effectiveness and a description of the output data.

(5) A list of all variable names used in the program with their definitions.

(6) A description of the statistical test procedures used and the results of these tests.

CHAPTER VI

AHS-1 A COMPUTER SIMULATION

AHS-1 is an event store computer simulation of an anti-submarine warfare helicopter small area search. In accordance with the definitions of Chapter I, it may properly be described as an analytic computer war game. A maximum of ten dipping sonar equipped helicopters and one evading submarine are the principles in the game.

The simulation is intended as a tool for the statistical analysis of the comparative effectiveness of different helicopter search tactics in similar tactical and physical environments. The model for AHS-1 and the underlying assumptions concerning sonar parameters and submarine capabilities are described in detail in the succeeding sections.

6.1 Tactical situation and play of the game.

A play of the game begins with the submarine at a known datum and a flight of sonar equipped helicopters enroute to or in the vicinity of the datum. The submarine leaves the surface at game time zero and departs the datum on a course and speed which is unknown to the helicopters. The helicopters arrive at datum at some time determined by the game user and proceed to dip stations in accordance with the assigned search plan.

The submarine leaves datum on a course, speed and depth selected in one of four ways according to the game inputs. Each of these operating modes reflects a different degree of submarine randomness, ranging from a completely predetermined

track to random selection of course, depth and speed within restrictions imposed by the game planner.

The helicopters proceed to their designated dip stations at datum time plus time late and commence the search. At each dip station as many as five sonar sweeps at various depths can be executed. Helicopters are assumed to be equipped with scanning type variable depth active sonar. The time to complete each sonar dip is determined by input time delays representing the following: (1) The time initially required to establish a hover and to lower the transducer, (2) the times necessary to change the transducer depth between sweeps, (3) the elapsed time from completion of the last sonar sweep until the helicopter has transitioned from hovering to forward flight, and (4) time spent in the prosecution of non-submarine contacts if applicable. The dip cycle time is the sum of dip time and time enroute between dips. Except for delays resulting from non-submarine contacts, dip time is the same for each helicopter. All helicopters fly at the same airspeed between dips.

At the beginning of each sonar sweep the submarine position is determined and a target range computed. The detection area is doughnut shaped, centered about the helicopter's position. Detection will occur if the submarine is within the annulus determined by the maximum detection circle and a smaller circle within which detection cannot occur because the helicopter is too close to the target. This minimum detection range is not dependent on sonar conditions and may be input as zero if desired. The maximum detection range is computed for each sweep and each

helicopter at the beginning of the play and recomputed whenever the submarine changes depth. Detection ranges are based on a randomly selected figure of merit for each unit and the prevailing sonar conditions. Once selected, individual helicopter figure of merits are constant throughout each play.

The scheme for computing detection ranges was designed by R. L. Klinkner of the Applied Physics Laboratory [17] and is considered by the author of this thesis to be an important improvement over the deterministic "cookie-cutter" detection range employed by many sonar simulations. The actual detection range for each helicopter depends on sonar frequency, sea state, layer depth, temperature in the layer, target depth, transducer depth, and the quality of the sonar operator-equipment combination. The latter is assumed to vary among helicopters in a prescribed random manner.

The play proceeds until all helicopters have completed their last dip or a detection occurs. At the completion of the play the running tally of detections is brought up to date and a new play commenced. This sequence continues until the desired number of plays have been completed. Input parameters and variables may then be changed and another sequence repeated. A series of plays with any given set of input data will be referred to as a game run.

6.2 Submarine movement.

The target submarine maneuvers in a three dimensional playing area. Horizontal movement is relative to an X,Y coordinate system, with yards the basic unit of distance. Depths are

measured in feet to the nearest foot. Courses are to the nearest degree and speeds to the nearest knot. Position and speed vectors are computed using a standard polar coordinate system; however, courses and speeds are adjusted so that the Y axis corresponds to north and all courses are measured clockwise from it.

Four different modes of submarine operation are available to the user of AHS-1 as follows:

Option (1). Submarine track is predetermined by the user. This mode might be used when studying the effect of evasive submarine maneuvers on a particular helicopter search plan. If the probability of delay due to non-submarine contacts (see Section 6.4) is input as zero, the location of the helicopters will be known at all times to the game planner. Any level of intelligence concerning helicopter movement may then be attributed to the submarine. Similarly, dip times may be made essentially random by assigning positive probability to non-submarine contacts. In this way the submarine may be placed in the situation of knowing where the helicopters are at any one time but not knowing when they will move, or where they will jump to.

Option (2). Speed, depth and the bearings of each track leg relative to the first course are predetermined as in Option (1). However, at the beginning of each play a pseudo-random number is selected from the interval $[0,360)$ degrees and added to each course. Election of this mode is equivalent to assuming a complete lack of knowledge, by the submarine, concerning helicopter movement. As opposed to a completely random submarine (Option 3), however, the capability of changing course, depth and/or speed

during the play is retained.

Randomizing the submarine track serves to preclude accidental biasing of game results by the user. This bias may be introduced by a particularly fortuitous selection of the submarine track with respect to the helicopter positions. A situation particularly suited to this mode is that of a submarine departing datum on a straight course but decreasing speed at the end of preset time intervals. This tactic might be used by a submarine commander who desires to open the range to datum as expenditiously as possible without exhausting the submarine batteries.

Option (3). Submarine course, speed and depth are uniformly distributed random variables. Course is distributed between zero and 360 degrees. Depth and speed range between upper and lower limits chosen by the game user. Course, speed, and depth are determined by generating three pseudo-random numbers at the beginning of each play, and remain the same until the termination of that play.

Option (4). Course and speed are chosen randomly as in Option (3) but depth is uniquely determined by the chosen speed and the game inputs. After selecting a speed, the minimum depth is taken as the shallowest depth at which the submarine can operate without cavitating. It should be noted here that cavitation does not directly affect the possibility of detection in the game, as the helicopters do not conduct passive sonar operations. This submarine mode is introduced as a means of restricting the target to depths which are realistic for the speed used. Target

depth is a parameter which is considered in determining active sonar range.

6.3 Helicopter movement.

Helicopter movement consists of jumps between sonar dip stations. All helicopters jump at the same speed, and as the game is now programmed dip stations are deterministic. Provisions have been made in the model and accompanying computer code for the introduction of random bearing and course errors. However, determination of the distribution of such errors would require the services of fleet units not available to the writer. With the increasing sophistication of helicopter navigation equipment the omission of these errors is not considered to be critically detrimental to the purpose of this simulation. The provision for including navigation errors is intended primarily as an area for further study by the interested reader.

Helicopter movement over the playing area is unrestricted in azimuth, and dip stations are determined by the game user. The game is primarily intended to simulate close area search plans, and helicopter dip stations are computed relative to datum. When simulating screening or other support operations the datum coordinates may represent an aircraft carrier or other helicopter take-off point. The first submarine maneuver may then be used to move the submarine from datum to the desired starting point. This maneuver can be executed at a high enough speed so that it will be essentially instantaneous.

6.4 Submarine detection.

Each helicopter has the capability of making up to five sonar

sweeps at each dip station. Transducer depth and the time required to adjust transducer depth and conduct a search are input by the user and may be different for each sweep. The number of sweeps, sensor depth and time to complete corresponding sonar sweeps are the same for all helicopters.

No provision is made for submarine motion during a sonar sweep. This is of very little consequence if scanning type sonar is simulated, but could have some effect on the outcome if searchlight sonar is being used. In the case of searchlight type sonar, the time to train the transducer may be included in each sonar sweep time; however, range to the submarine will only be determined at the beginning of each sweep, regardless of the time required to step train the transducer through a complete sweep.

At the beginning of a sonar sweep the range to the submarine is computed according to

$$Rs = \sqrt{(X_h - X_s)^2 + (Y_h - Y_s)^2} \qquad (6.1)$$

where X_h and Y_h are the dip coordinates of the helicopter, and X_s and Y_s are the submarine position coordinates. Detection occurs if $SR \leq Rs \leq Rd$ where Rd is the detection range for the particular helicopter at the appropriate transducer depth, Rs is the target range, and SR is the minimum range at which detection can occur. SR may be input as zero if desired.

The course, speed and depth of the submarine at the beginning of a dip are used in computing target range. If the target is scheduled to maneuver during any sweep this information is noted and at the completion of the sweep in question the dip is

discontinued. The submarine maneuver is then executed and the dip completed using the correct submarine track information.

In addition to sonar sweep times and delays representing the time required to lower and retrieve the sonar transducer, one other factor contributes to sonar dip time. This is a random variable representing time engaged in pursuing non-submarine contacts.

For the purposes of this game the time required to classify actual submarine targets is not considered pertinent. If a submarine has been detected, the search plan has been effective and classifying the target is another problem. It is conceivable, however, that the effectiveness of different search plans may be more or less sensitive to variation in dip times among helicopters. One of the primary factors contributing to differences in dip time is the classification of false contacts. In the model it has been assumed that non-submarine targets such as fish or whales are uniformly distributed throughout the ocean. This implies that one sonar operator has about the same chance of contacting such a target as any other sonarman. Once a non-submarine contact has been generated, the time required to classify it as non-submarine will vary among sonar operators. These two assumptions have been incorporated into the model in the following way.

The actual delay for each helicopter is dependent on two input parameters, the probability of obtaining a false contact (P_{fc}) and the maximum time any non-submarine contact will be

prosecuted (Tmax). A random number from the interval [0, 1] is first generated to determine whether the helicopter gained a false contact during the dip in question. If this random number is less than or equal to P_{fc} a non-submarine contact was acquired and a delay time must be added to dip time. The length of time to be added is determined by

$$T = \frac{RN}{P_{fc}} \cdot Tmax \qquad (6.2)$$

Where T is the actual delay and RN is the previously generated random number.

6.5 Sonar detection ranges.

Detection ranges for each helicopter are computed at the beginning of every play and whenever the submarine changes depth during the play. As detection range depends on transducer depth, every helicopter will have associated with it as many different detection ranges as the number of sonar sweeps per dip. The actual computation of these detection ranges is a problem in acoustics rather than simulation and will not be considered in detail. The interested reader is referred to Fundamentals of SONAR, by J. W. Horton [19] for further information on this subject.

The basic equation for active sonar is:

$$EE = FM + TS - 2PL \qquad (6.3)$$

where EE is the echo excess (signal level relative to that required for a 50% probability of detection)

FM = Sonar figure of merit

TS = Target strength

PL = One-way propagation loss

All quantities in equation (6.3) are expressed in decibels. Figure of merit and target strength are input variables. Figure of merit is a measure of the quality of the sonar operator-equipment combination independent of water conditions. Target strength is a characteristic of the submarine size and shape, the external surface of the submarine hull, and target aspect. An average value of target strength is used in the model. An echo excess of zero corresponds to a 50% probability of detection and this value is used in the computation.

In computing sonar ranges it is assumed that sonar figure of merit varies among helicopters according to some known probability distribution. It is also assumed that figure of merit does not vary appreciably between dips for the same helicopter. Therefore the figure of merit is computed for each helicopter only once for each play of the game. In order that this thesis be unclassified no attempt has been made to duplicate the actual distribution of helicopter figure of merit. For purposes of AHS-1 a Normal (gaussian) distribution is assumed with mean and variance as input parameters.

Rewriting equation (6.3) with echo excess as zero:

$$PL = \frac{1}{2} (FM + TS) \qquad (6.4)$$

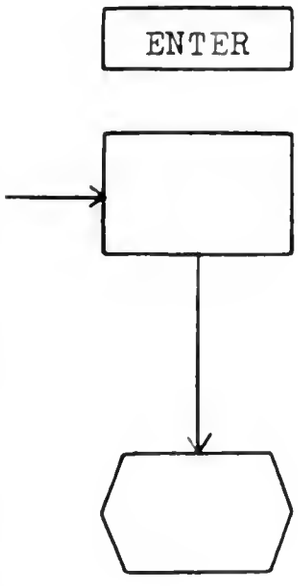
Propation loss (PL) combined with transducer depth and the remaining sonar parameters, is used by the routine which computes detection ranges for each individual helicopter. The other

input variables entering into detection range are target depth, sonar frequency, layer depth, temperature in the layer and sea state.

6.6 Description of program subroutines.

The following brief descriptions of the individual subroutines are intended to show how the model described above has been implemented. A secondary purpose of these descriptions is to indicate auxiliary functions performed by each event subroutine. Rules for input data card preparation are included under Subroutine PRINT. Flow charts showing the logical structure of each subroutine are included at the end of the individual descriptions. the Following flow chart symbols are used throughout.

Flow chart symbols. The meaning of each symbol is determined by shape and letters within the symbol. Differences in size reflect only space consideration.

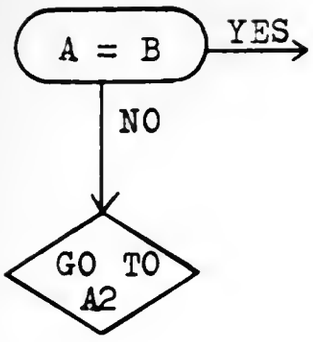


- Beginning point in a subroutine.

- Computation to be carried out within the subroutine.

- Arrows show direction of flow.

- Computation to be carried out by a subroutine other than the one in which the symbol appears. In the detailed flow charts the information needed by the routine being called is written inside the block.



- Decision point in the logic. The route to be followed is indicated by the YES or NO.

- A logical jump is to be made to some point within the subroutine.

- Beginning point of a subsection of logic within a subroutine. A logical jump will always go to such a point. The numeral indicates page number.

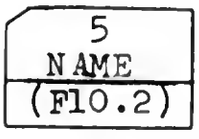
- Continuation of the logical flow to the next page. This symbol would appear at the bottom of page one and at the top of page two.

- End of event subroutine. Program is to process the next event.

- End of an event subroutine when for any reason the normal program logic is to be interrupted.

- Logical end of any subroutine which is not an event routine. Indicates a jump back to the section of the program that called the subroutine into action.

- Instruction to stop the game. Indicates that all the data has been processed or a mistake was made in the input information.



- Data is to be read into the machine. In the detailed flow charts the top numeral indicates the tape unit to read from and the characters in parentheses indicate the format to be used. NAME is the variable name to be read.

- Write output on tape unit six. The numeral at the bottom is a code indicating the format.

- Store New Event. Type of event and event time will appear within the block.

Symbols used within blocks in detailed charts.

RN	- Random Number
*	- Multiplication
/	- Division
**	- Exponentiation
A+B → C	- Add A to B and store the answer in the variable name C.

List of variable names. Following is a list of the variable names with their definitions. The names are arranged according to the subroutine in which they were originally defined. This also corresponds to the order of the COMMON statements. Input data names are designated by an asterisk. Names representing data to be printed are identified by $ or # symbols. Dimensioned variables are indicated by parentheses following the name.

(1) Utility Words

I T1 T2	Used for temporary storage or convenience throughout the program.
IDIP I2	Helicopter dip counters. I2 is set equal to the total number of dips (number of helicopters times number of dips per helicopter) at the beginning of each play. IDIP is incremented each time a helicopter dips. If a detection does not occur the play is over when IDIP = I2.

J2	Counter for number of samples. Used in computing average number of detections per sample.

(2) Main Control Program

*NRSAM	Number of samples in each simulation run.
*NSSIZE (I)	Number of plays constituting the I^{th} sample.
$NHSDET (I)	Counter for keeping track of the number of detections made during one sample by the $_{I}$th helicopter.
$TMIN	Minimum time in minutes from the time the helicopters arrive at datum until a detection occurred for current sample.
$TMAX	Maximum time required for a detection during any sample.
$BART	Average time elapsed between time helicopters arrive at datum and detection when detection occurs.
$VART	Sample variance of detection time when detection occurs.
#MIND	Minimum number of plays resulting in detections during a series of runs.
#MAXD	Maximum number of plays resulting in detections during a series of runs.
#BARD	Average number of plays resulting in detections per sample during a series of runs.

$Printed after each sample is completed.

#Printed at the end of runs designated by game user.

VARD — Sample variance of number of detections per sample.

$ PERC — Percentage of plays resulting in a detection during a sample.

* DAT — Time at which helicopters arrive at datum and begin search.

* XDAT
* YDAT — X and Y coordinates of datum.

$ NDET — Counter which is set to zero at beginning of each sample and incremented each time a detection occurs.

NDTEMP — Temporary counter which is set equal to NDET at the beginning of each play. Comparison of NDET and NDTEMP signals whether the play ended because a detection occurred or because the helicopters completed all dips.

PL(I) — One way sonar propagation loss of the I^{th} helicopter. Computed at the beginning of each play.

* IRNC — Number of pseudo-random numbers to be generated and disposed of at the beginning of a simulation run.

(3) Subroutines SNE and TNE

The following three words are used throughout the program to transfer information to or from the Event Store Table.

TIMET — Time at which an event is to be stored or executed.

NREVT — Number of the event to be stored or executed.

NRUNT — Number of the unit affected by the event.

NTNE — Counter which keeps track of the location of the last event in the table.

TIME(I)
NREV(I)
NRUN(I) — These three arrays comprise the event store table, and represent time of execution, number of the event subroutine to be called, and the number of the unit involved in the event.

(4) Subroutine SME (Submarine Maneuver Event)

	KT	Submarine maneuver counter.
*	NRANSS	Signals whether submarine maneuvers are predetermined or random.
*	NRANC	Signals whether the first course is randomly selected when submarine maneuvers are predetermined.
	STIME	Time the submarine last maneuvered.
	SMNEXT	Time at which the submarine is scheduled to make the next maneuver.
	DT	Depth at STIME.
	VXS	V_x at time STIME.
	VYS	V_y at time STIME.
	XS YS	X and Y coordinates at time STIME.
	TT	Temporary storage for random numbers which are to be added to submarine courses.

Submarine maneuver table.

*	SMTIME(I)	Time to execute the I^{th} submarine maneuver.
*	DEPTH(I)	Depth (feet) at time SMTIME(I).
*	SCUS(I)	Course (degrees) at time SMTIME(I).
*	SSPD(I)	Submarine speed (knots) at time SMTIME(I).
*	NRMAN	Number of times the submarine is to maneuver when maneuvers are predetermined.

Input parameters when submarine maneuvers are random.

*	NHISPD LOSPD	Upper and lower limits (knots) of submarine speed.
*	MAXSD MINSD	Upper and lower limits (feet) of submarine depth.

Cavitation speed versus depth table.

* NCAVS(I) Cavitation speed in knots for the I^{th} point on the curve.

* NCAVD(I) Minimum depth corresponding to NCAVS(I).

* NCAV Number of entries in the cavitation table.

(5) Subroutine HSM (Helicopter Maneuver Event).

IDIPN(I) Counter which keeps track of the number of dips made by the I^{th} helicopter.

* NRDIPS Number of dips to be made by each helicopter.

* HTE(I,J) Jump time of I^{th} helicopter enroute to the J^{th} dip station. Input as yards and converted to minutes internally.

* HBRG(I,J) Bearing relative to the preceding leg flown by the I^{th} helicopter in transiting to the J^{th} dip station. Input as degrees and converted to radians internally.

XH(I)
YH(I) X and Y coordinates of the I^{th} helicopters last dip station.

* NRHS Number of helicopters.

* HSPD Speed in knots at which helicopters transit between dip stations.

HSPDT HSPD converted to yards per minute.

* TDD Time (minutes) required to establish a hover and lower the transducer.

ERB Error in bearing added to HBRG(I,J) to determine actual flight path of helicopters.

ERT Distance error added to HTE(I,J).

XHT(I)
YHT(I) Table for temporary storage of helicopter X and Y coordinates. Used whenever ERB and ERT are not computed.

* NTE Signals whether ERB and ERT are to be computed. Value is one if errors are to be computed, and zero otherwise.

(6) Subroutine HDE (Detection Event)

	DRNG(I,J)	Detection range of the I^{th} helicopter for the J^{th} sonar sweep.
*	NRSW	Number of sonar sweeps to be made by each helicopter at each dip station.
*	SWT(I)	Time (minutes) required to complete the I^{th} sonar sweep.
*	PD(I)	Transducer depth for the I^{th} sonar sweep.
*	TRD	Time (minutes) required to retrieve the transducer at the end of each dip.
	NOVER	Signals that a play has been completed. Value is one if play is over, zero otherwise.
*	SHORTR	Range (yards) from helicopter within which target is too near for detection to occur.
	NSWP(I)	Signals the first sweep number to be executed.
*	PRFC	Probability of detecting a non-submarine contact.
*	FCTMAX	Maximum time for evaluation of non-submarine contacts.
	FCT	Actual time to evaluate non-submarine contact.

(7) Subroutine COMPRG

*	NS	Sea State. (beaufort scale).
*	T	Temperature in the layer. (degrees Fahrenheit).
*	F	Acoustic frequency of the helicopter sonar.
	DP	Transducer depth. (feet).
*	DL	Layer depth. (feet).
*	TS	Target strength. (decibels).
*	FOMMU	Average figure of merit of the helicopter sonar. (decibels).
*	FOMVAR	Variance of helicopter sonar figure of merit. (decibels).

Main Program. The main AHS-1 program performs the functions of moderator and bookkeeper. This section of the program calls for more input data when it is needed and records the results of each play. The main program also sets up the initial submarine and helicopter maneuver events so that each replay of the game will begin at the proper point in time and space.

The following terms will be referred to throughout the description of the program.

Play. A play of the game begins when the submarine leaves datum at game time zero and ends either when the submarine is detected by a helicopter or when all helicopters have completed their last dip.

Sample. A sample consists of a series of plays. Sample size is determined by the user and is ordinarily based on what is considered necessary for statistical validity of the results.

Run. A run consists of one or more samples.

At the completion of a run, any number of input parameters may be changed. Inputs do not vary between samples of the same run, additional samples representing replications of the same experiment. The only difference between two samples of the same run is the series of random numbers used. The random number generator is reset to the first number in the series at the beginning of each run.

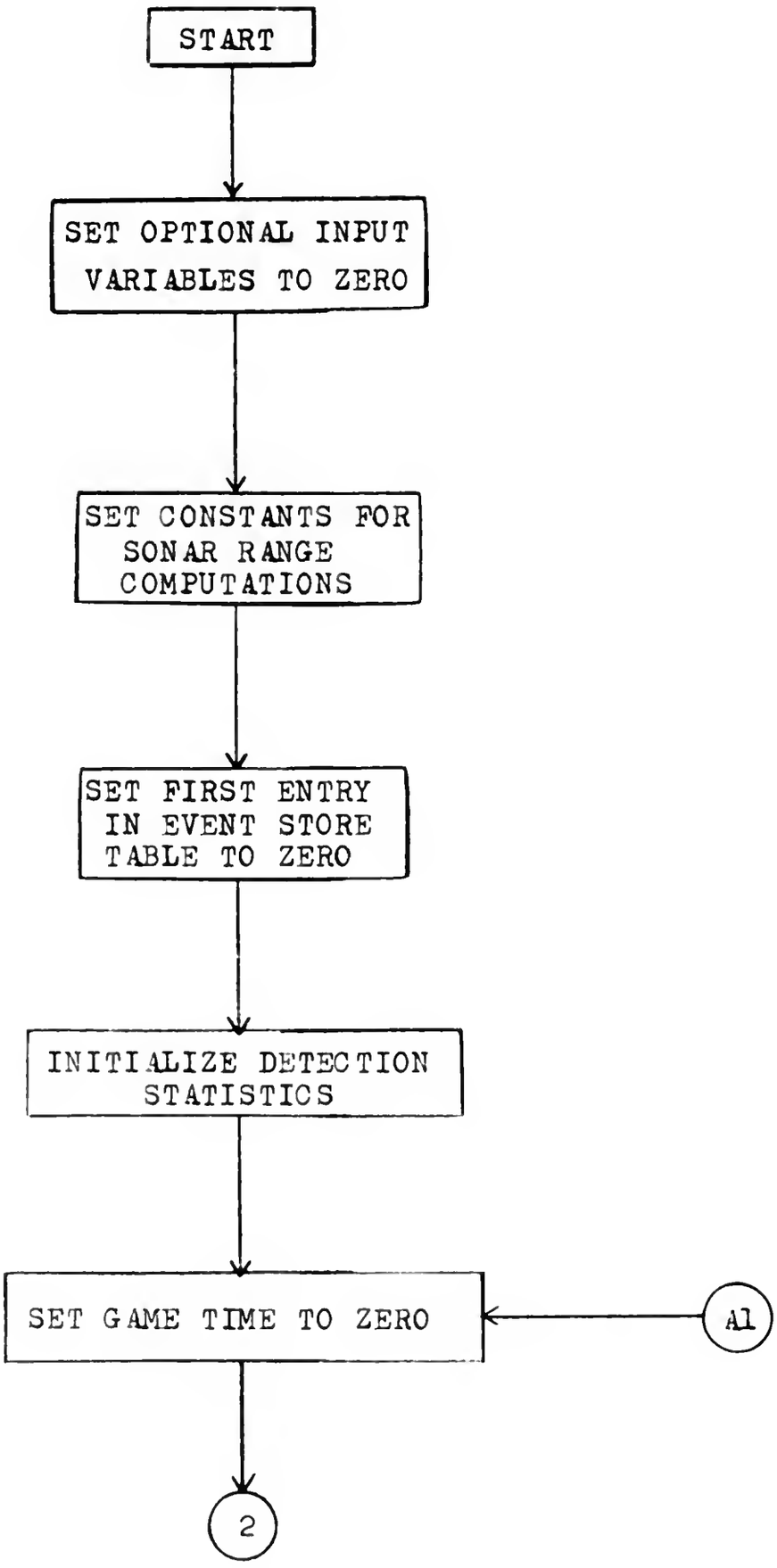
The various functions of the main program should be apparent from the accompanying flow diagram once the reader is familiar with each individual subroutine. Only those parts which are not immediately clear will be mentioned here.

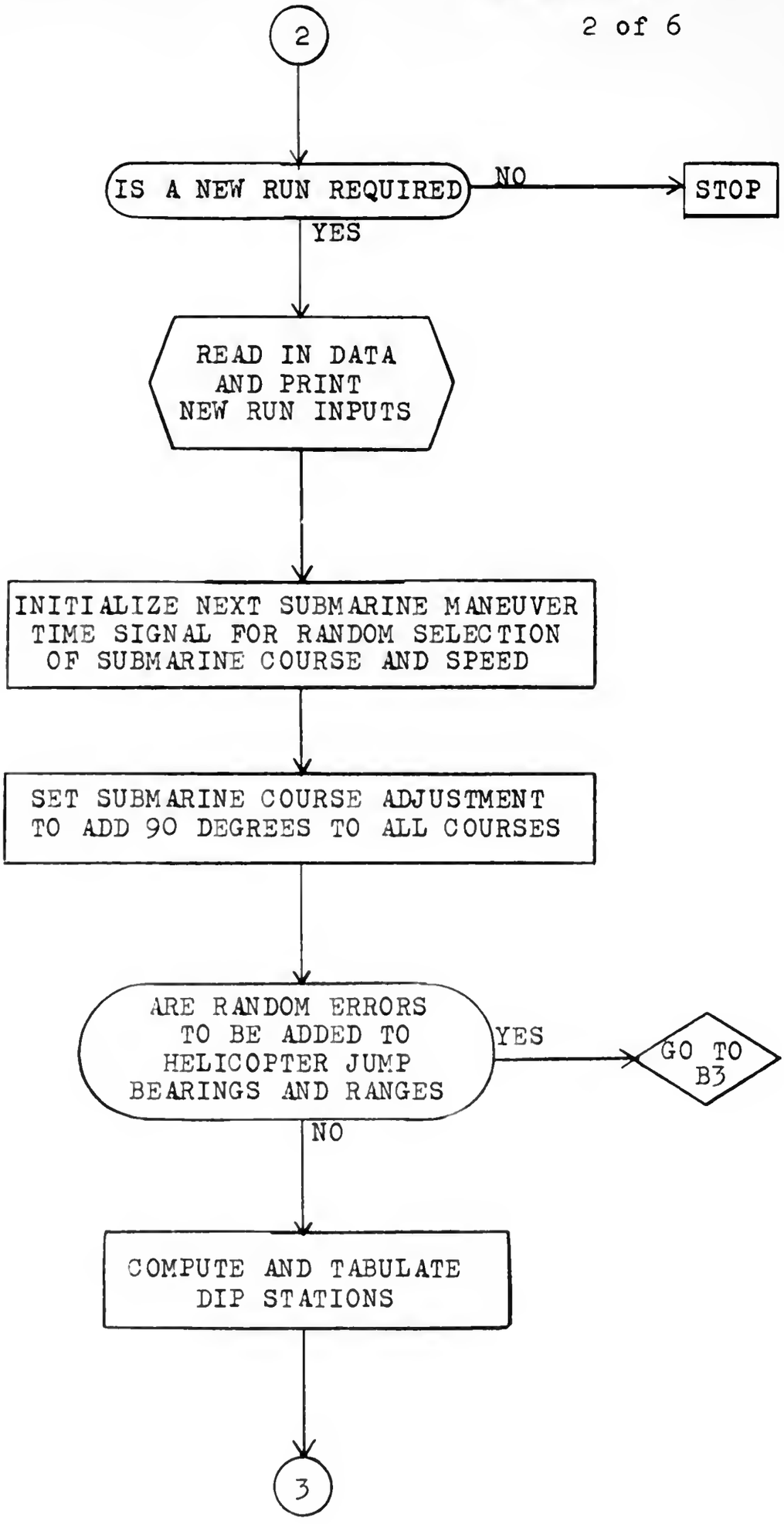
It is contemplated that the usual submarine operating mode will be a completely random submarine, option (3). The datum coordinates will normally be the origin of the X,Y, coordinate system. For this reason variable names associated with these inputs, referred to as optional variables in the flow chart, are preset prior to the first run. This obviates the necessity of reading these inputs in except when other than this standard operating mode is desired.

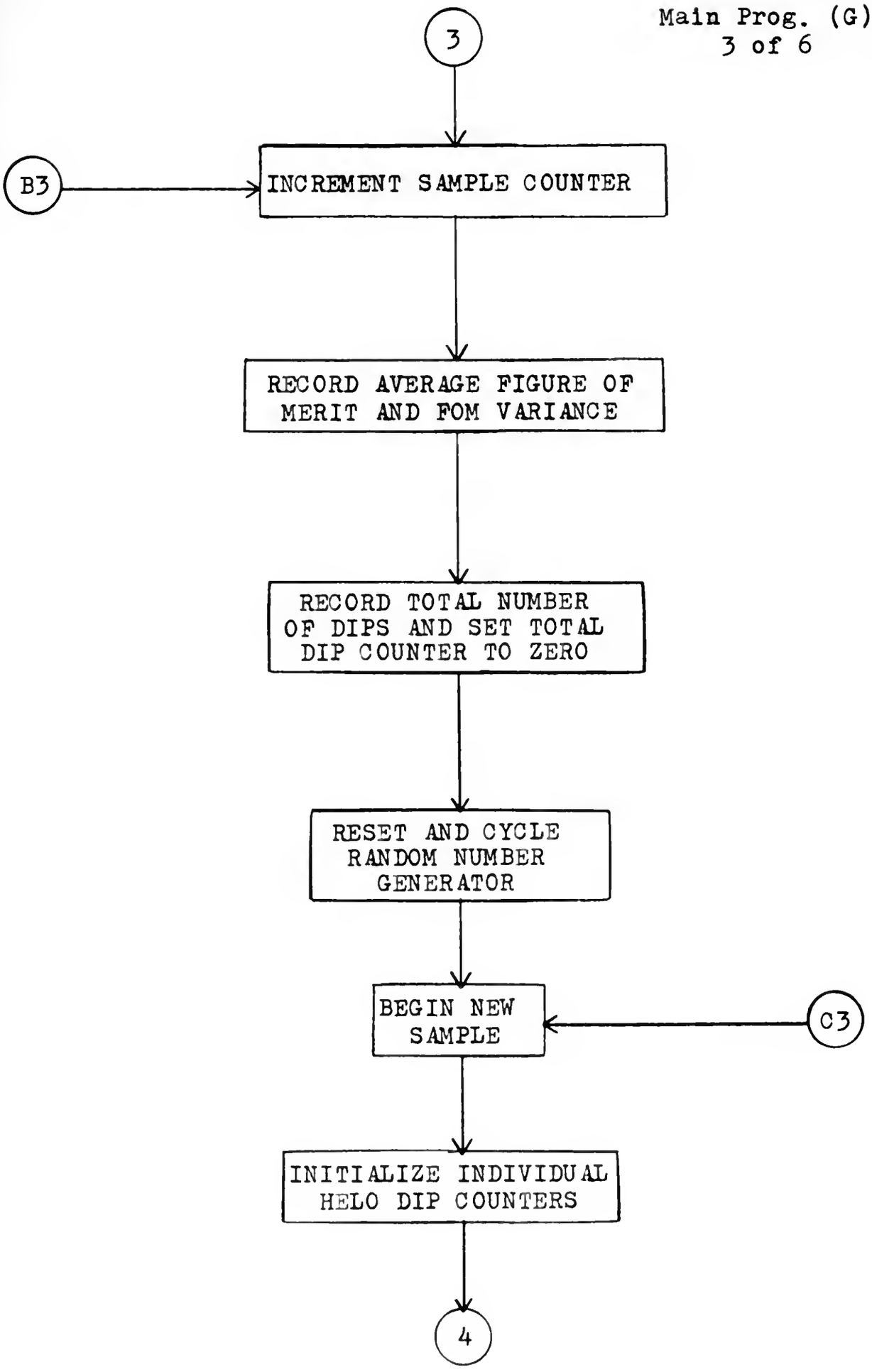
The minimum number of detections and maximum time to detect are initially set at large numbers so they will always be replaced by the actual quantities once a detection occurs. If the submarine is not detected these qauantities will be printed as 1000 detections and 600 minutes respectively and should be disregarded. If a sample size of 1000 or larger is used, the main program should be changed to reflect this by setting MIND equal to a number larger than the sample size.

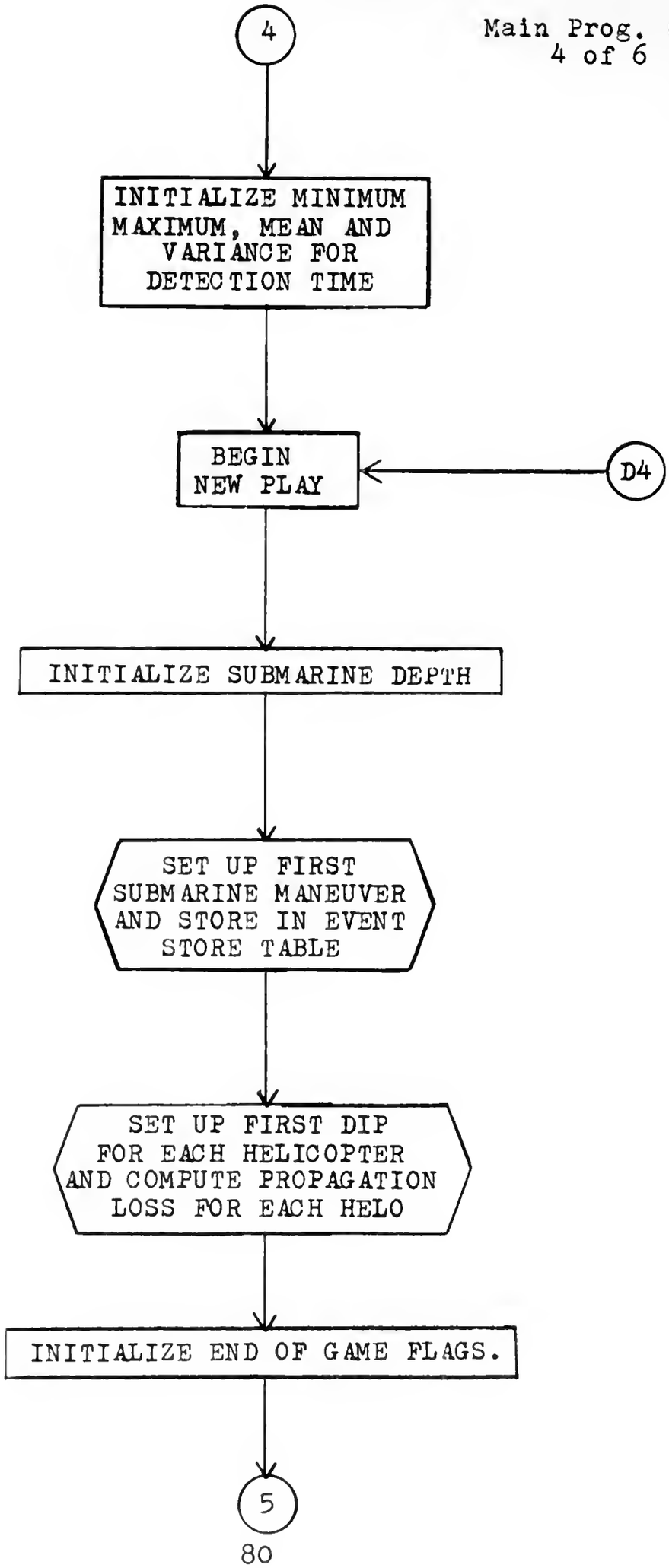
Bearing and course inputs to the program are measured clockwise, courses being relative to north, or 000. For computational convenience the zero radial is congruent with the X axis and all bearings are measured counter-clockwise. For purposes of the game itself this presents no difficulty since only relative distances and bearings affect the simulation results. With no adjustment the resulting game would be a mirror image of what one would plot from the inputs, and relative distances would be preserved. In anticipation that the user might wish to have unit positions printed out at some time during the play, an adjustment has been made to all courses. By using the negative of

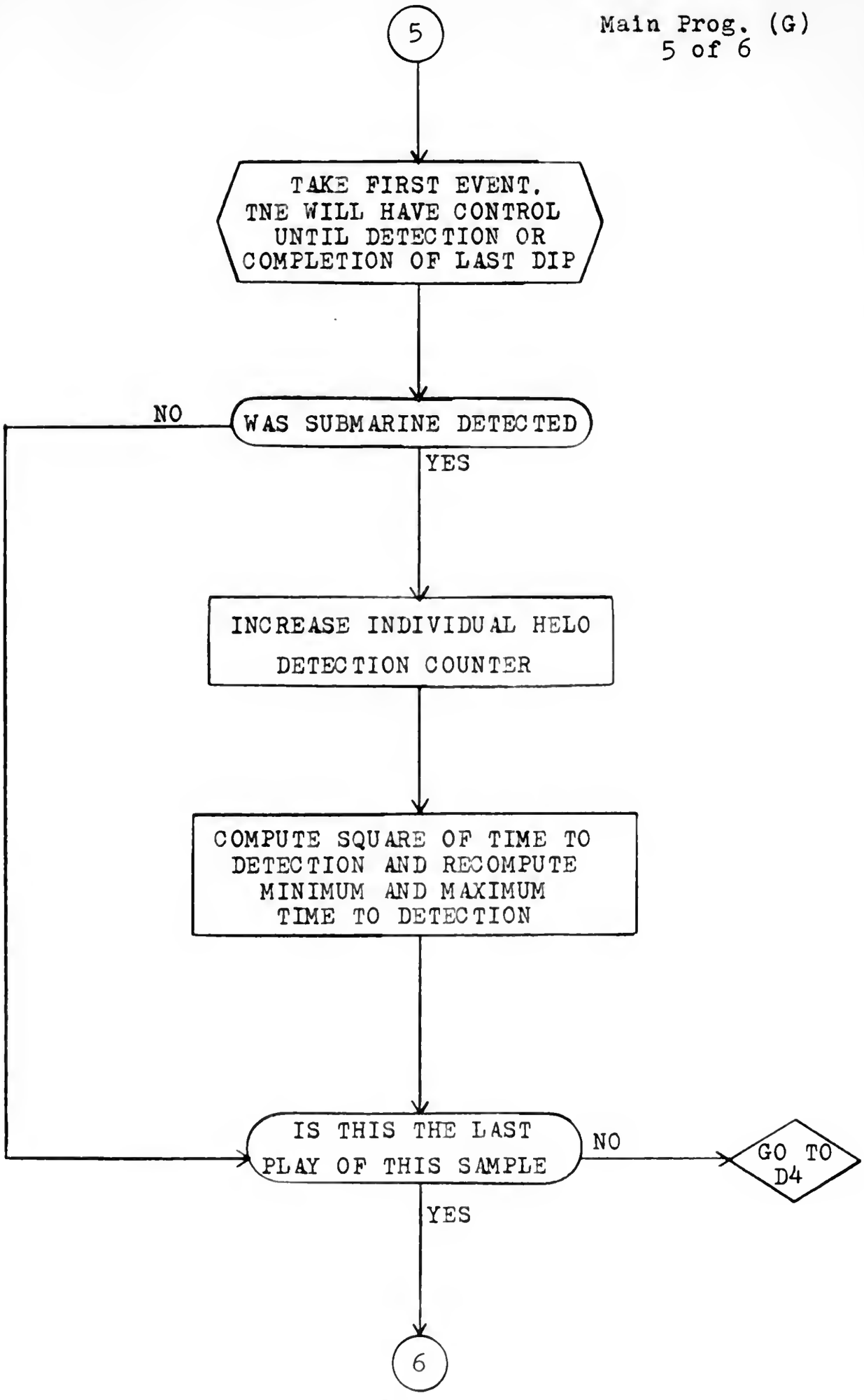
courses as input to the program and adding ninety degrees, the coordinate system has been transformed into a geographical map. Consequently submarine and helicopter positions will agree with the usual map representation.

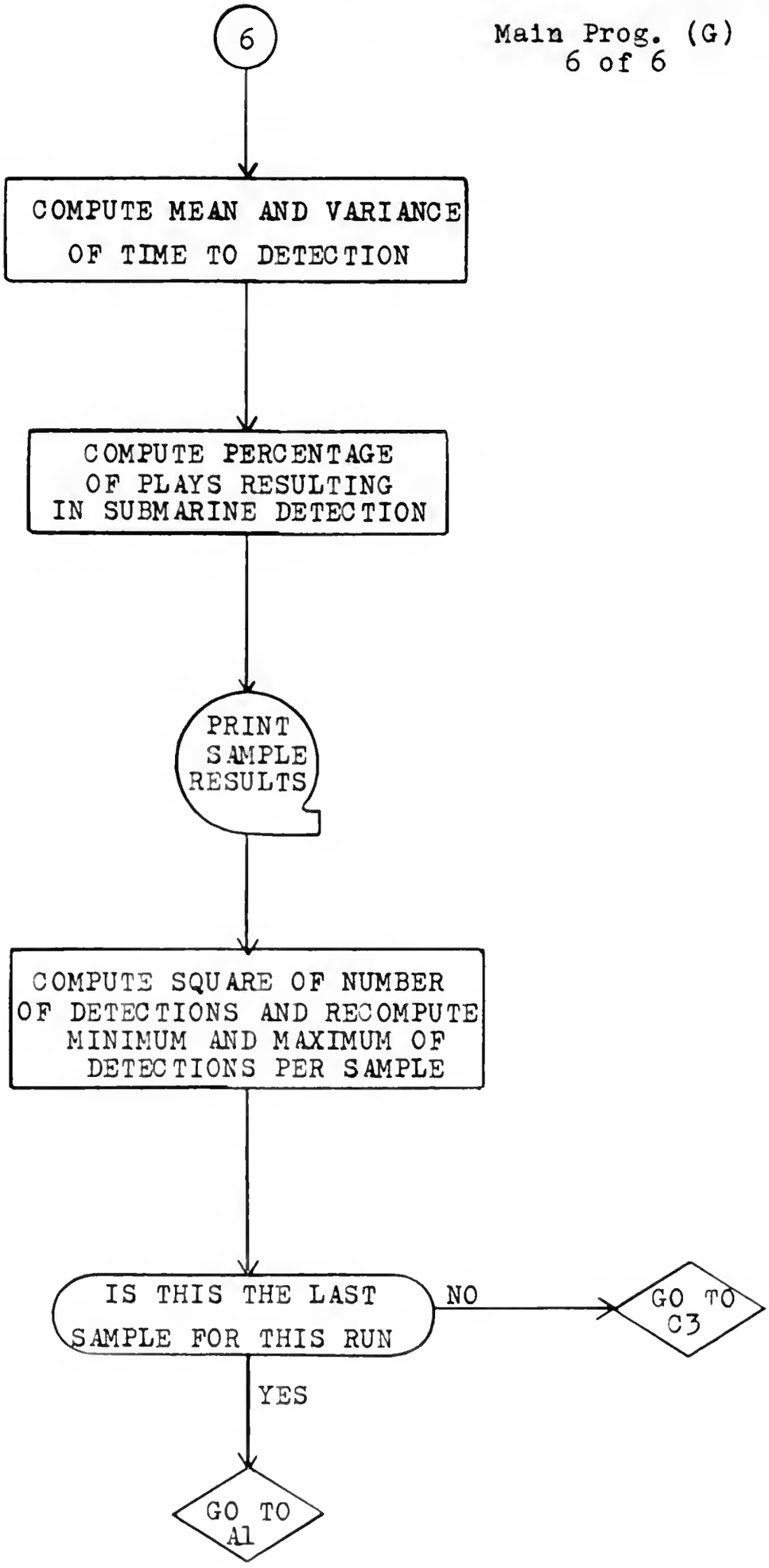

START
SET OPTIONAL INPUT VARIABLES TO ZERO
SET CONSTANTS FOR SONAR RANGE COMPUTATIONS
SET FIRST ENTRY IN EVENT STORE TABLE TO ZERO
INITIALIZE DETECTION STATISTICS
SET GAME TIME TO ZERO
A1
2


2
IS A NEW RUN REQUIRED
NO
STOP
YES
READ IN DATA AND PRINT NEW RUN INPUTS
INITIALIZE NEXT SUBMARINE MANEUVER TIME SIGNAL FOR RANDOM SELECTION OF SUBMARINE COURSE AND SPEED
SET SUBMARINE COURSE ADJUSTMENT TO ADD 90 DEGREES TO ALL COURSES
ARE RANDOM ERRORS TO BE ADDED TO HELICOPTER JUMP BEARINGS AND RANGES
YES
GO TO B3
NO
COMPUTE AND TABULATE DIP STATIONS
3


3
B3
INCREMENT SAMPLE COUNTER
RECORD AVERAGE FIGURE OF MERIT AND FOM VARIANCE
RECORD TOTAL NUMBER OF DIPS AND SET TOTAL DIP COUNTER TO ZERO
RESET AND CYCLE RANDOM NUMBER GENERATOR
BEGIN NEW SAMPLE
C3
INITIALIZE INDIVIDUAL HELO DIP COUNTERS
4


4
INITIALIZE MINIMUM MAXIMUM, MEAN AND VARIANCE FOR DETECTION TIME
BEGIN NEW PLAY
D4
INITIALIZE SUBMARINE DEPTH
SET UP FIRST SUBMARINE MANEUVER AND STORE IN EVENT STORE TABLE
SET UP FIRST DIP FOR EACH HELICOPTER AND COMPUTE PROPAGATION LOSS FOR EACH HELO
INITIALIZE END OF GAME FLAGS.
5


5
TAKE FIRST EVENT. TNE WILL HAVE CONTROL UNTIL DETECTION OR COMPLETION OF LAST DIP
NO
WAS SUBMARINE DETECTED
YES
INCREASE INDIVIDUAL HELO DETECTION COUNTER
COMPUTE SQUARE OF TIME TO DETECTION AND RECOMPUTE MINIMUM AND MAXIMUM TIME TO DETECTION
IS THIS THE LAST PLAY OF THIS SAMPLE
NO
GO TO D4
YES
6

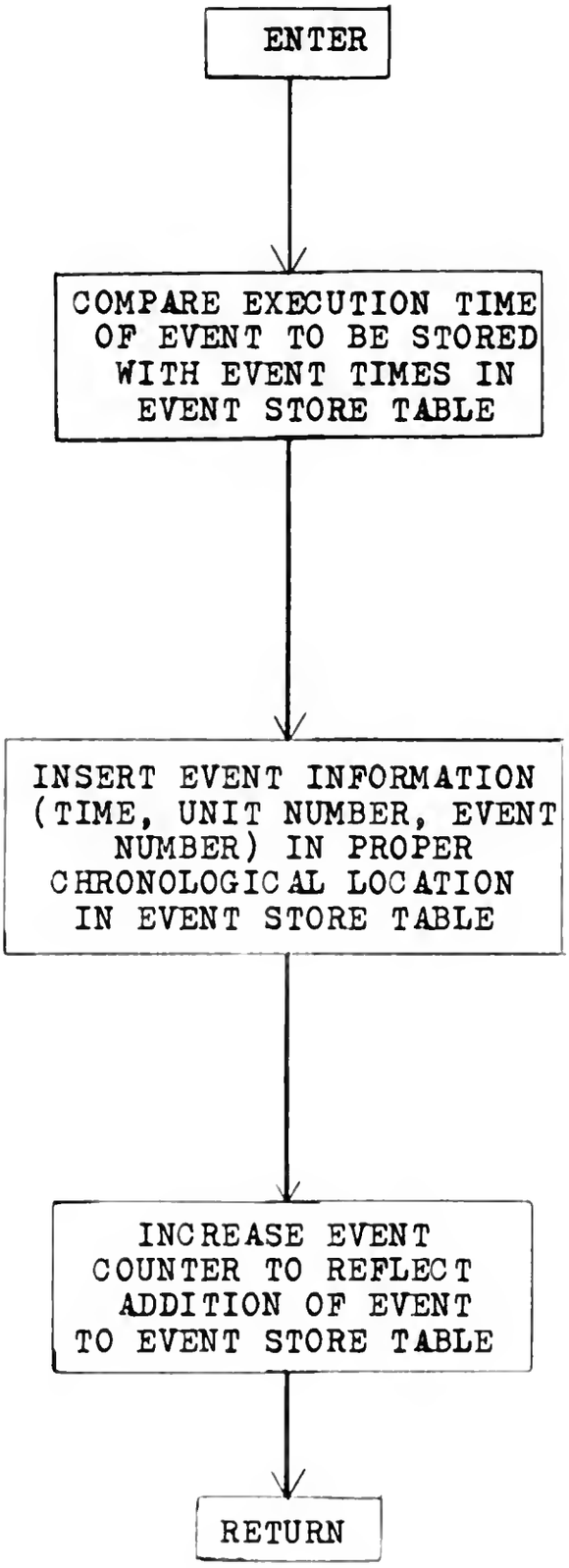

6
COMPUTE MEAN AND VARIANCE OF TIME TO DETECTION
COMPUTE PERCENTAGE OF PLAYS RESULTING IN SUBMARINE DETECTION
PRINT SAMPLE RESULTS
COMPUTE SQUARE OF NUMBER OF DETECTIONS AND RECOMPUTE MINIMUM AND MAXIMUM OF DETECTIONS PER SAMPLE
IS THIS THE LAST SAMPLE FOR THIS RUN
NO
GO TO C3
YES
GO TO A1

Subroutine SNE. Subroutine SNE (Store New Event) enters future events in chronological order in the event store table. The event store table is a list, ordered according to time, of actions which are to take place in the course of a play of the game.

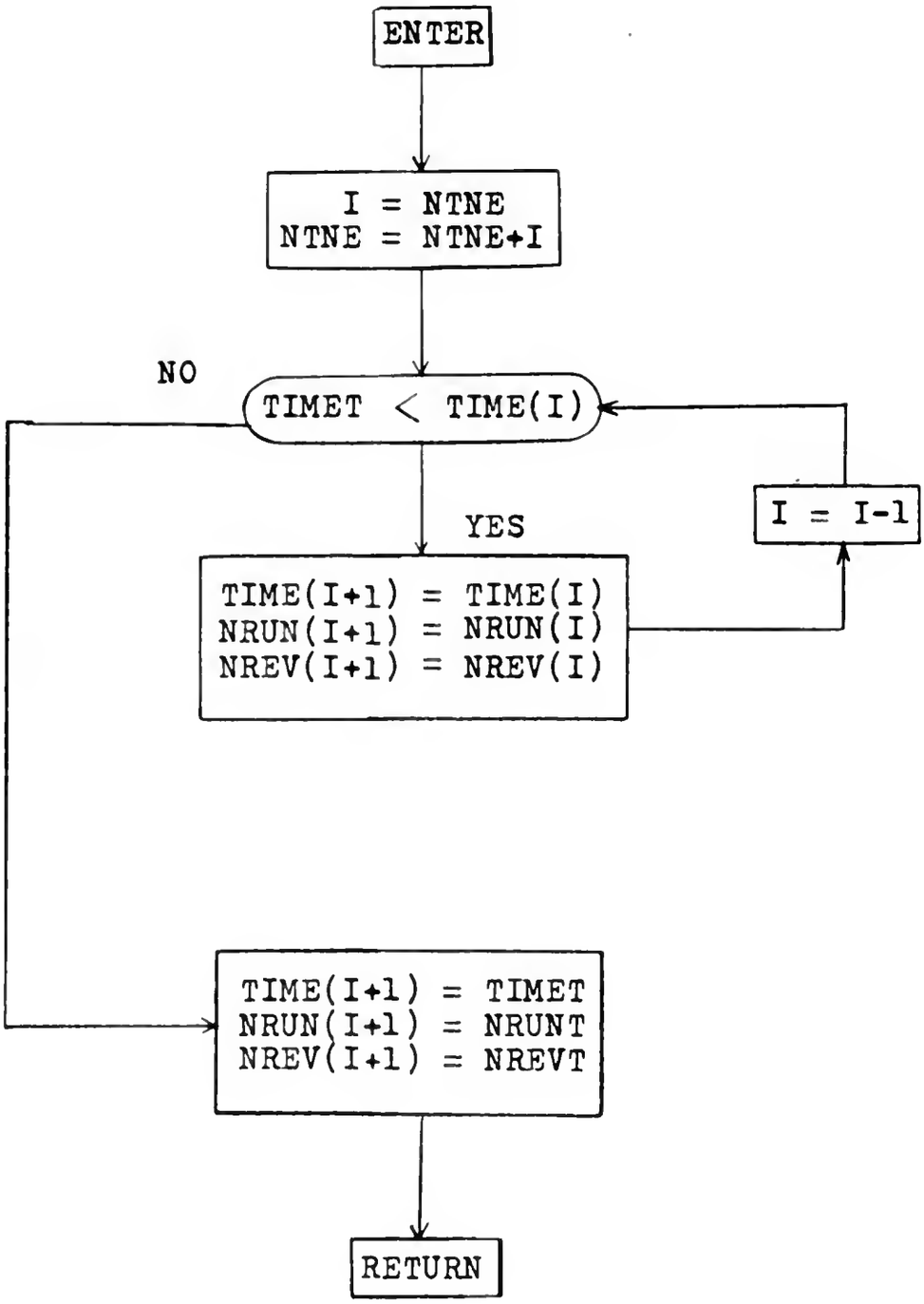
When an event is to be stored, SNE is entered with three items of information; the time at which the event is to occur, the type of event to be executed, and the number of the unit involved. This information constitutes one event word.

Should two or more events be scheduled to occur at the same time there is no designated ordering according to event type. Two or more such events will take place in the order in which they were processed by SNE.

Subroutine SNE (General)



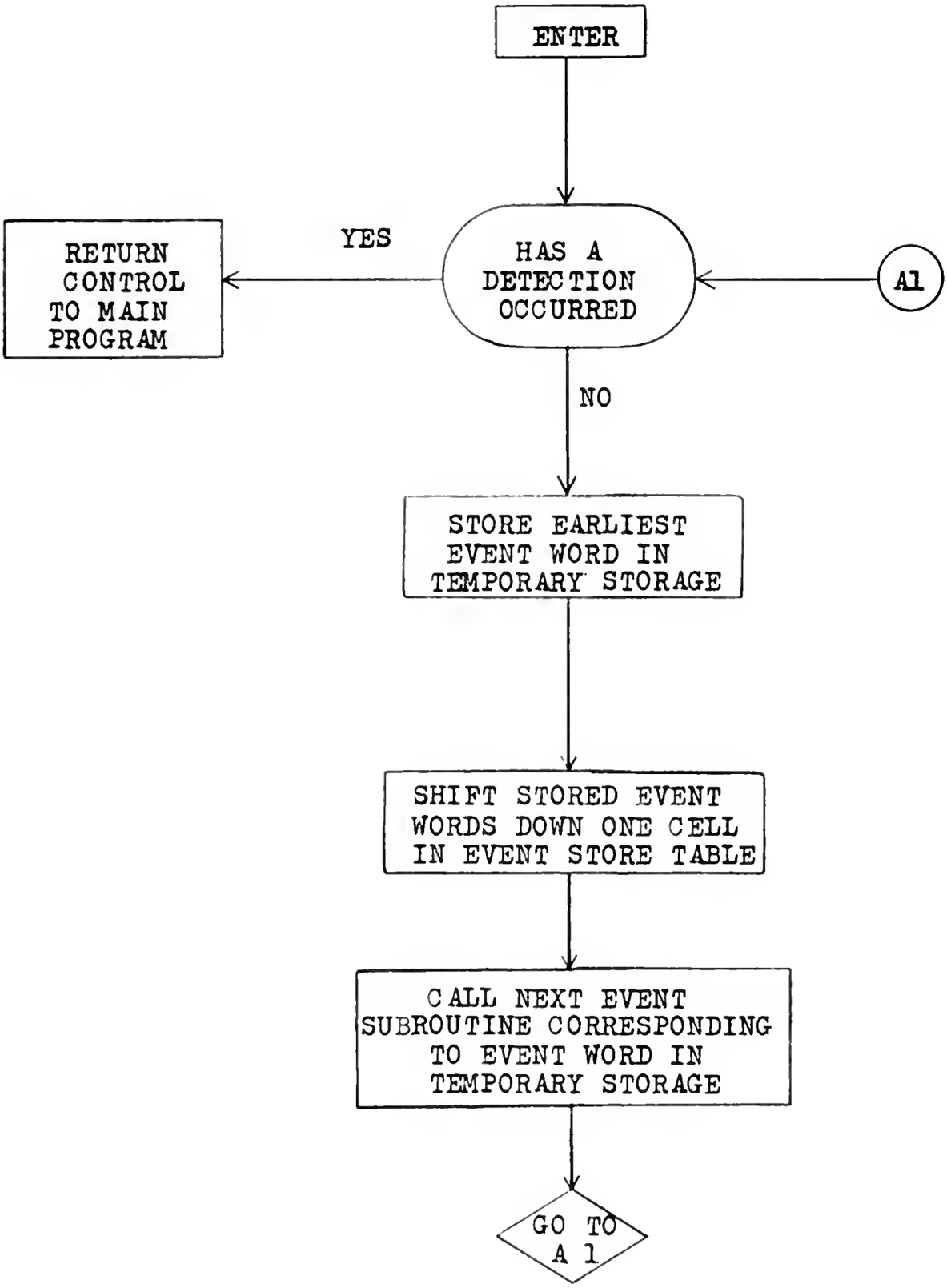
Subroutine SNE (Detailed)



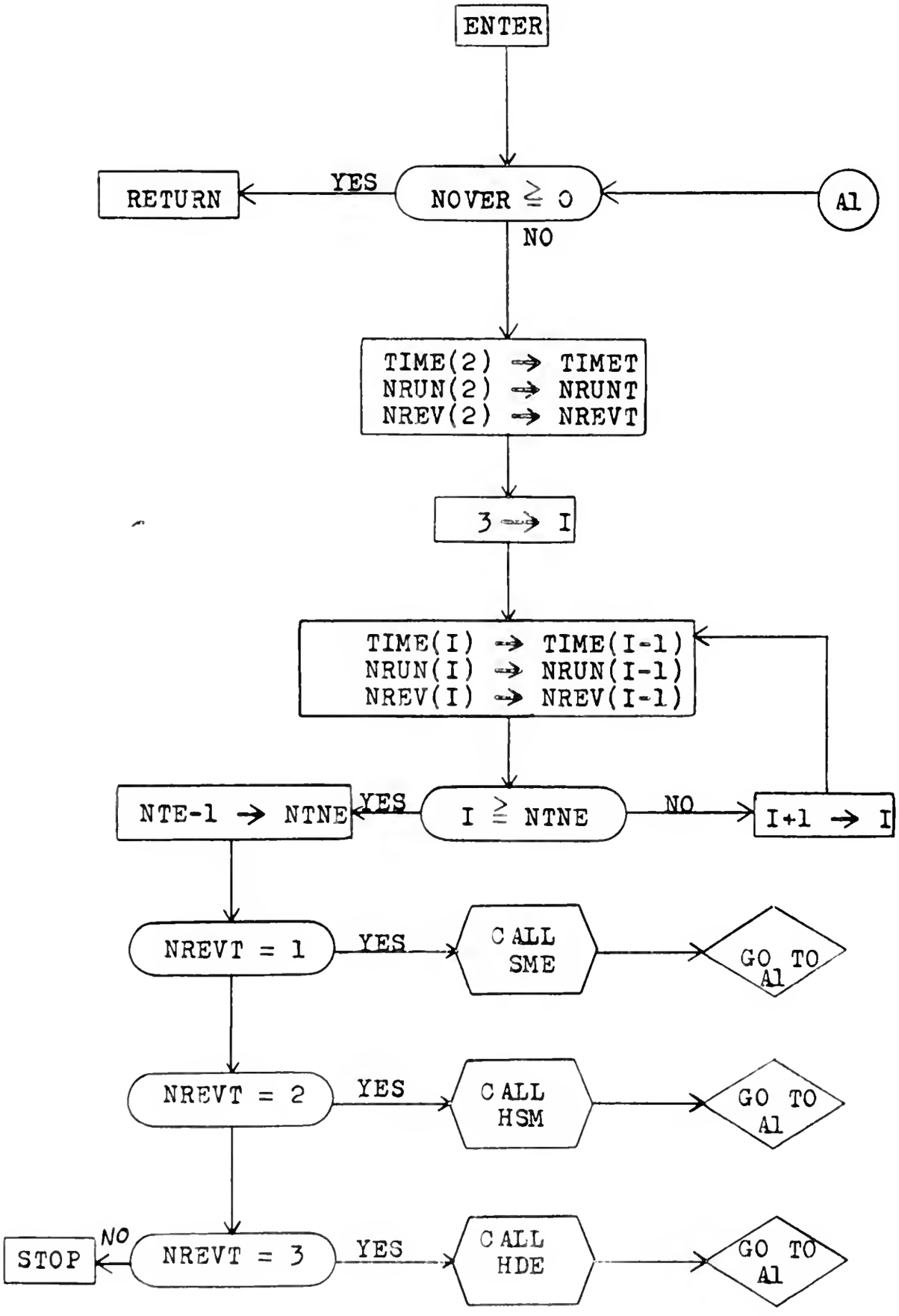
Subroutine TNE. Subroutine TNE (Take Next Event) examines the event type of the earliest entry in the event store table and calls the appropriate event subroutine into action. Once a play has commenced, the program is controlled by TNE until the end of the play. The end of a play is signaled by the detection event subroutine. When this signal is received, TNE transfers control back to the main program.

When an event is executed the event store entry referring to it is removed from the table. An alternative method would be to leave the entries in the table and keep track of the earliest unexecuted event. The method used is somewhat more time consuming but results in a considerable saving of storage space.

Subroutine TNE (General)



Subroutine TNE (Detailed)



Subroutine SME. The first time this subroutine (Submarine Maneuver Event) is called on during a play, the submarine operating mode is determined from the information input by the game planner. Once this determination has been made, course, speed and depth are determined randomly or read directly from the input information. Game time and the submarine position are noted and X and Y components of velocity are computed. From these four pieces of information the submarine's coordinate position can be determined at any later time.

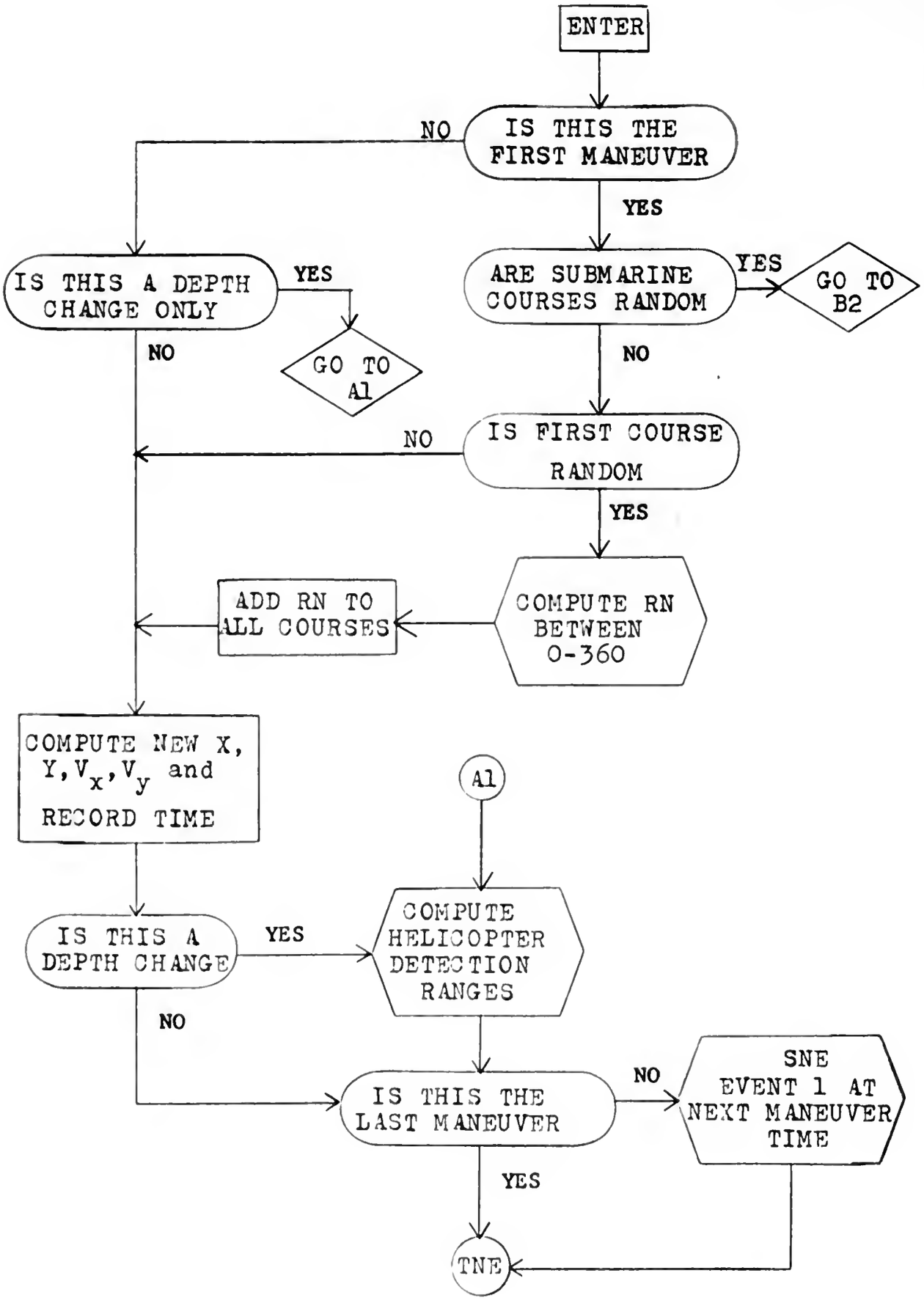
Additional functions performed by Subroutine SME are as follows:

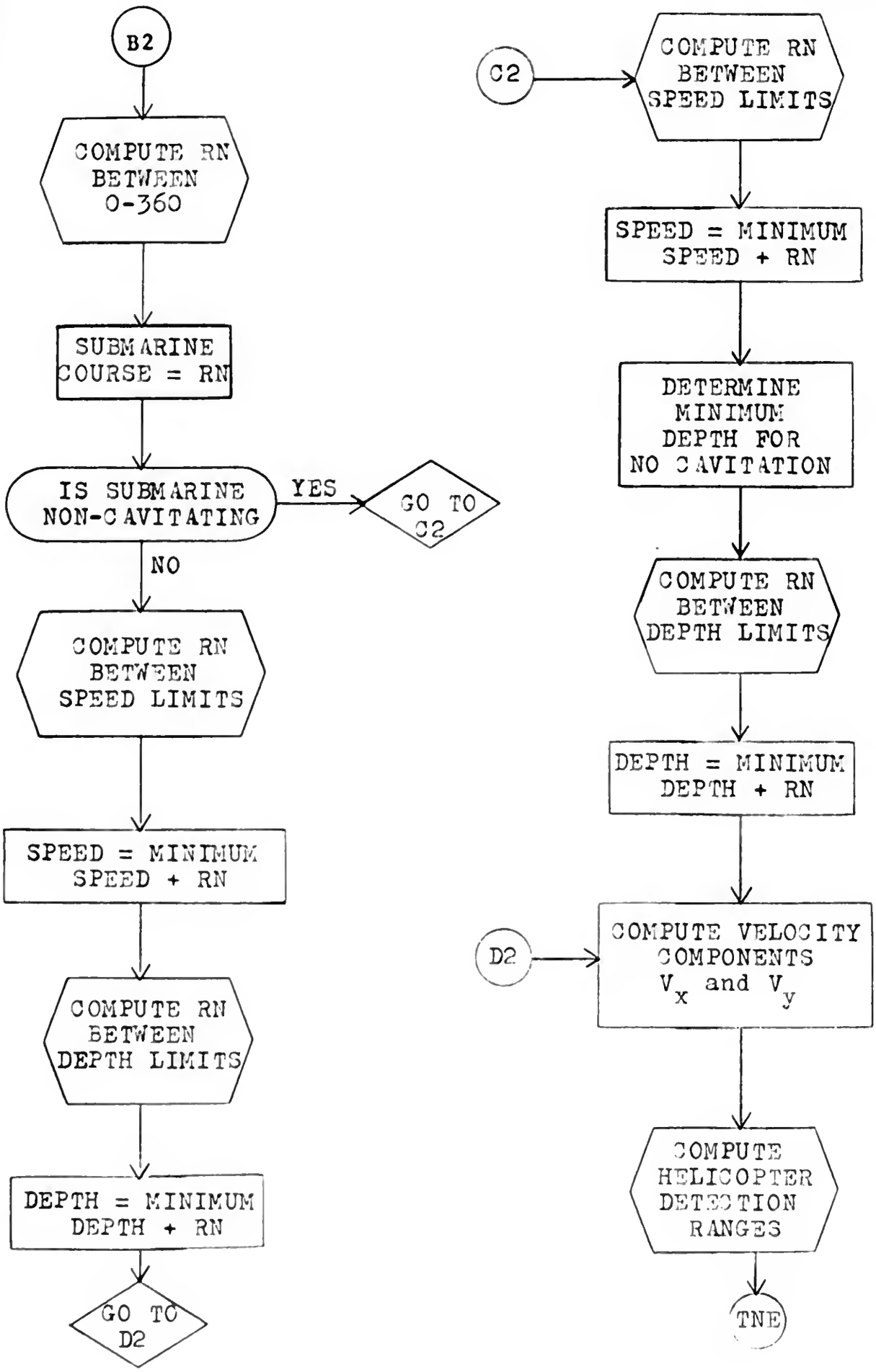
(1) On the first submarine maneuver or any subsequent maneuver involving a depth change, helicopter detection ranges are computed by calling Subroutine COMPRG.

(2) If another submarine maneuver is to take place, the time the maneuver is to occur is noted and subroutine SNE is instructed to place a submarine maneuver event in the event store table.

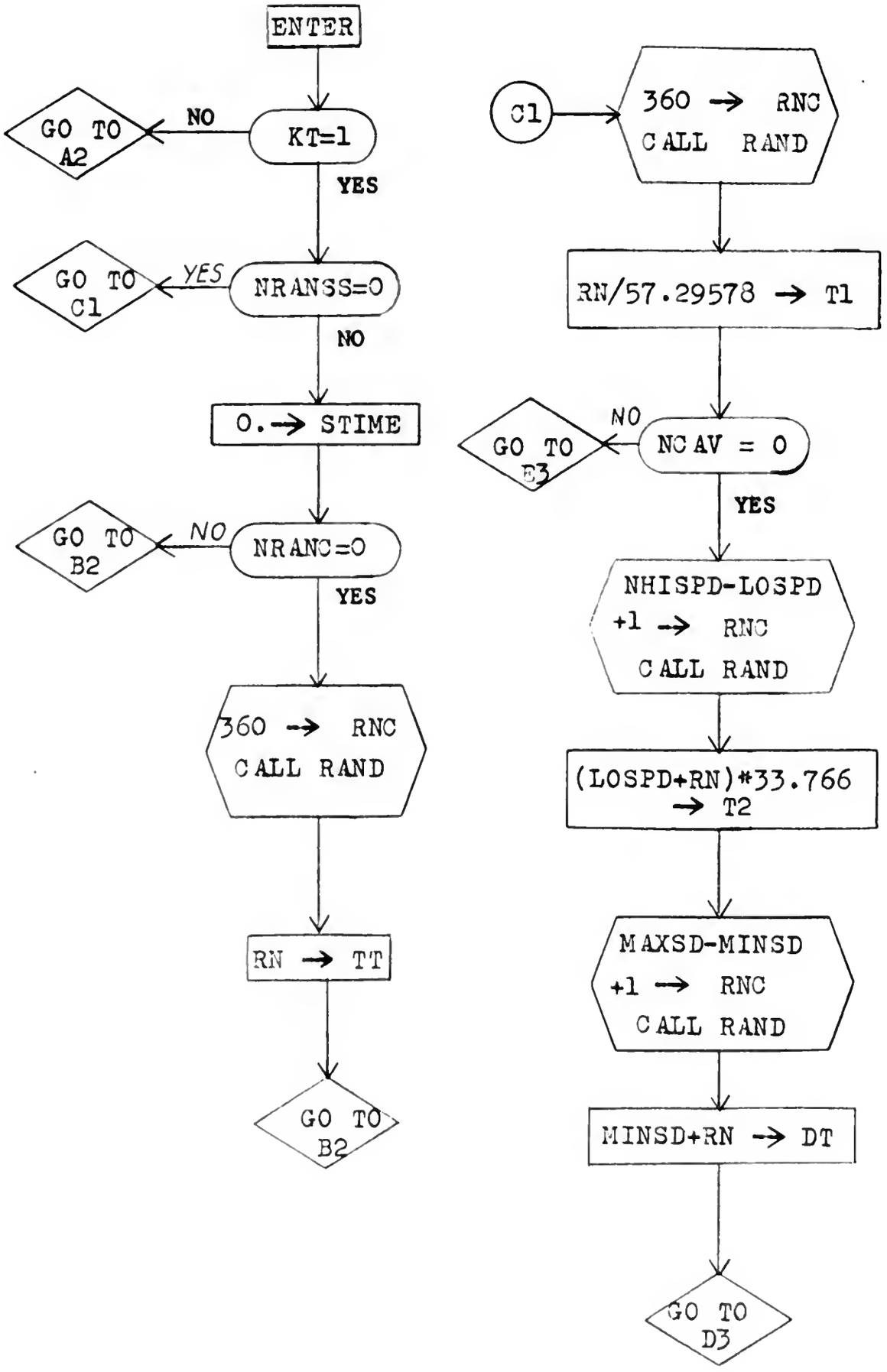
(3) The time of the next submarine maneuver is recorded for use by the detection event subroutine. This information is used in determining whether the submarine maneuvered during any helicopter dip. In the event of a submarine maneuver while a helicopter is dipping, submarine track and depth information must be updated before the dip is completed.

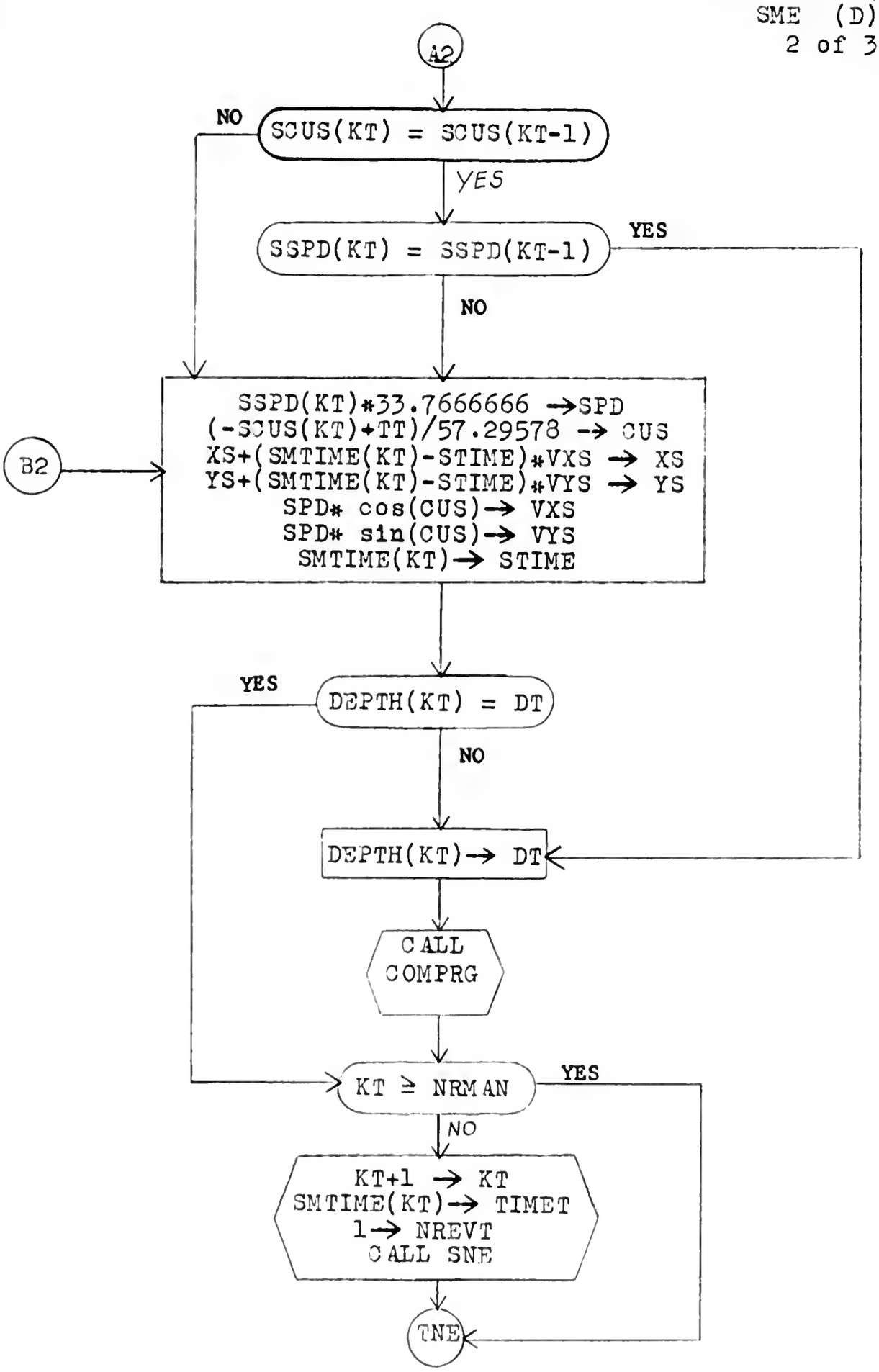
Subroutine SME – Submarine Maneuver Event (General)

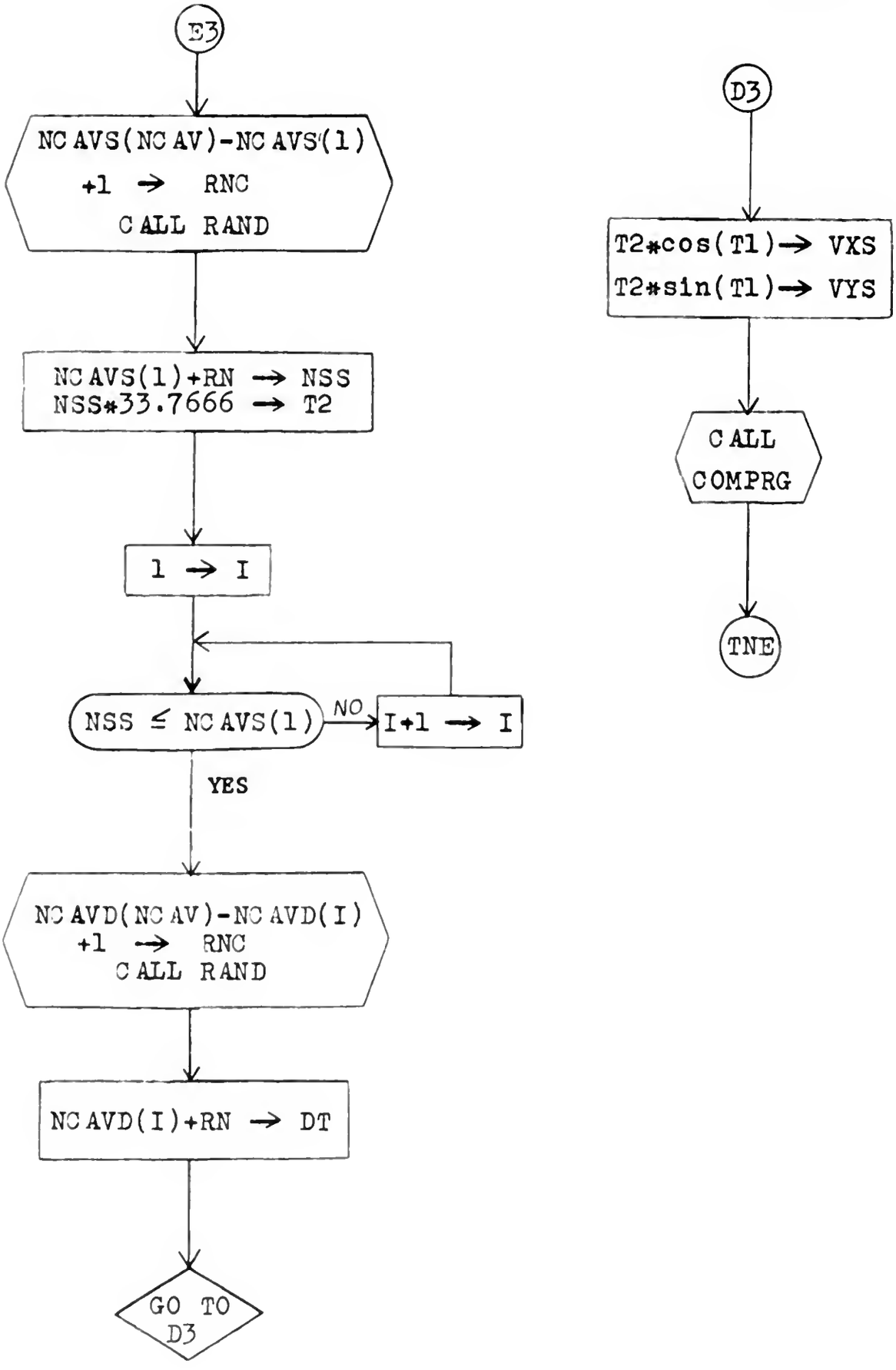



B2
COMPUTE RN BETWEEN 0-360
SUBMARINE COURSE = RN
IS SUBMARINE NON-CAVITATING
YES
GO TO C2
NO
COMPUTE RN BETWEEN SPEED LIMITS
SPEED = MINIMUM SPEED + RN
COMPUTE RN BETWEEN DEPTH LIMITS
DEPTH = MINIMUM DEPTH + RN
GO TO D2
C2
COMPUTE RN BETWEEN SPEED LIMITS
SPEED = MINIMUM SPEED + RN
DETERMINE MINIMUM DEPTH FOR NO CAVITATION
COMPUTE RN BETWEEN DEPTH LIMITS
DEPTH = MINIMUM DEPTH + RN
D2
COMPUTE VELOCITY COMPONENTS V_x and V_y
COMPUTE HELICOPTER DETECTION RANGES
TNE

Subroutine SME - Submarine Maneuver Event (Detailed)




A2
SCUS(KT) = SCUS(KT-1)
NO
YES
SSPD(KT) = SSPD(KT-1)
YES
NO
B2
SSPD(KT)*33.7666666 →SPD
(-SCUS(KT)+TT)/57.29578 → CUS
XS+(SMTIME(KT)-STIME)*VXS → XS
YS+(SMTIME(KT)-STIME)*VYS → YS
SPD* cos(CUS)→ VXS
SPD* sin(CUS)→ VYS
SMTIME(KT)→ STIME
DEPTH(KT) = DT
YES
NO
DEPTH(KT)→ DT
CALL
COMPRG
KT ≥ NRMAN
YES
NO
KT+1 → KT
SMTIME(KT)→ TIMET
1→ NREVT
CALL SNE
TNE


E3
NCAVS(NCAV)-NCAVS(1)
+1 → RNC
CALL RAND
NCAVS(1)+RN → NSS
NSS*33.7666 → T2
1 → I
NSS ≤ NCAVS(1)
NO
I+1 → I
YES
NCAVD(NCAV)-NCAVD(I)
+1 → RNC
CALL RAND
NCAVD(I)+RN → DT
GO TO D3
D3
T2*cos(T1) → VXS
T2*sin(T1) → VYS
CALL COMPRG
TNE

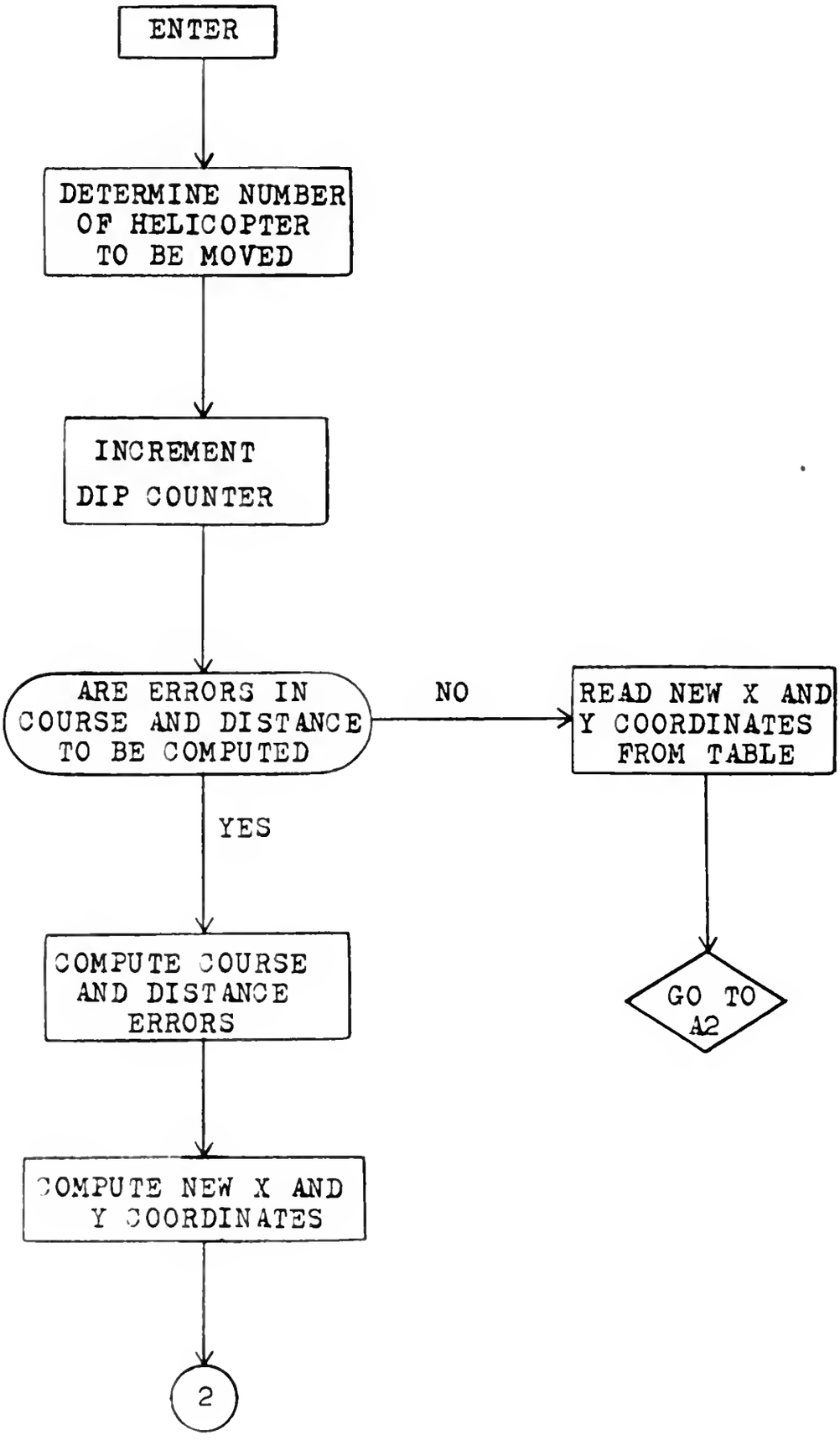
Subroutine HSM. Subroutine HSM (Helicopter Maneuver Event) updates each helicopter's dip station coordinates and records this information for use by the detection event subroutine. As the program is presently written, dip coordinates are computed by the main program and stored in a table. This table is available to Subroutine HSM whenever a helicopter is to be maneuvered. This requires that the computations be made only once for each game run. Should the necessary code for computing distance and bearing errors be written, new dip coordinates will be computed by Subroutine HSM every play.

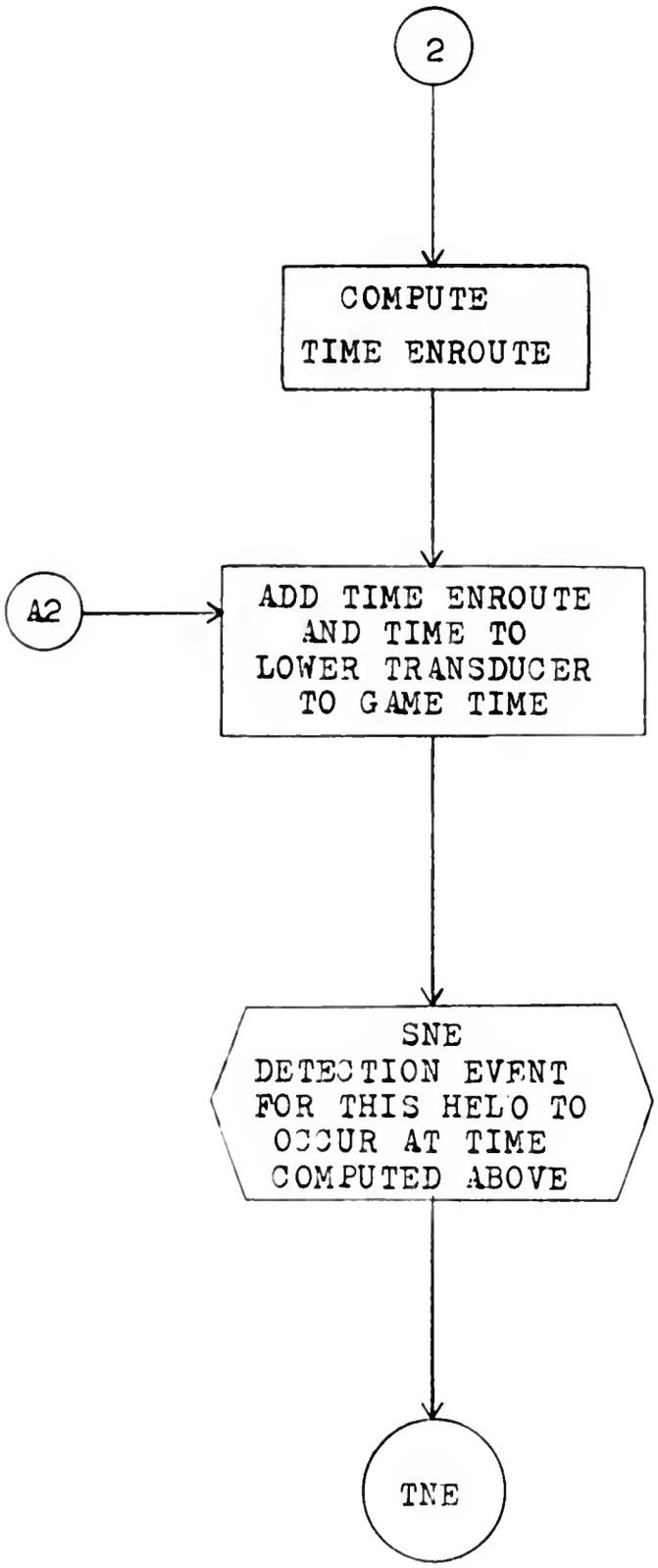
Additional functions of this subroutine are as follows:

(1) Records the number of the last dip station occupied by each helicopter. After the first dip, the order of maneuvering is determined by dip cycle time rather than helicopter unit number. Therefore an individual dip counter must be maintained for each helicopter.

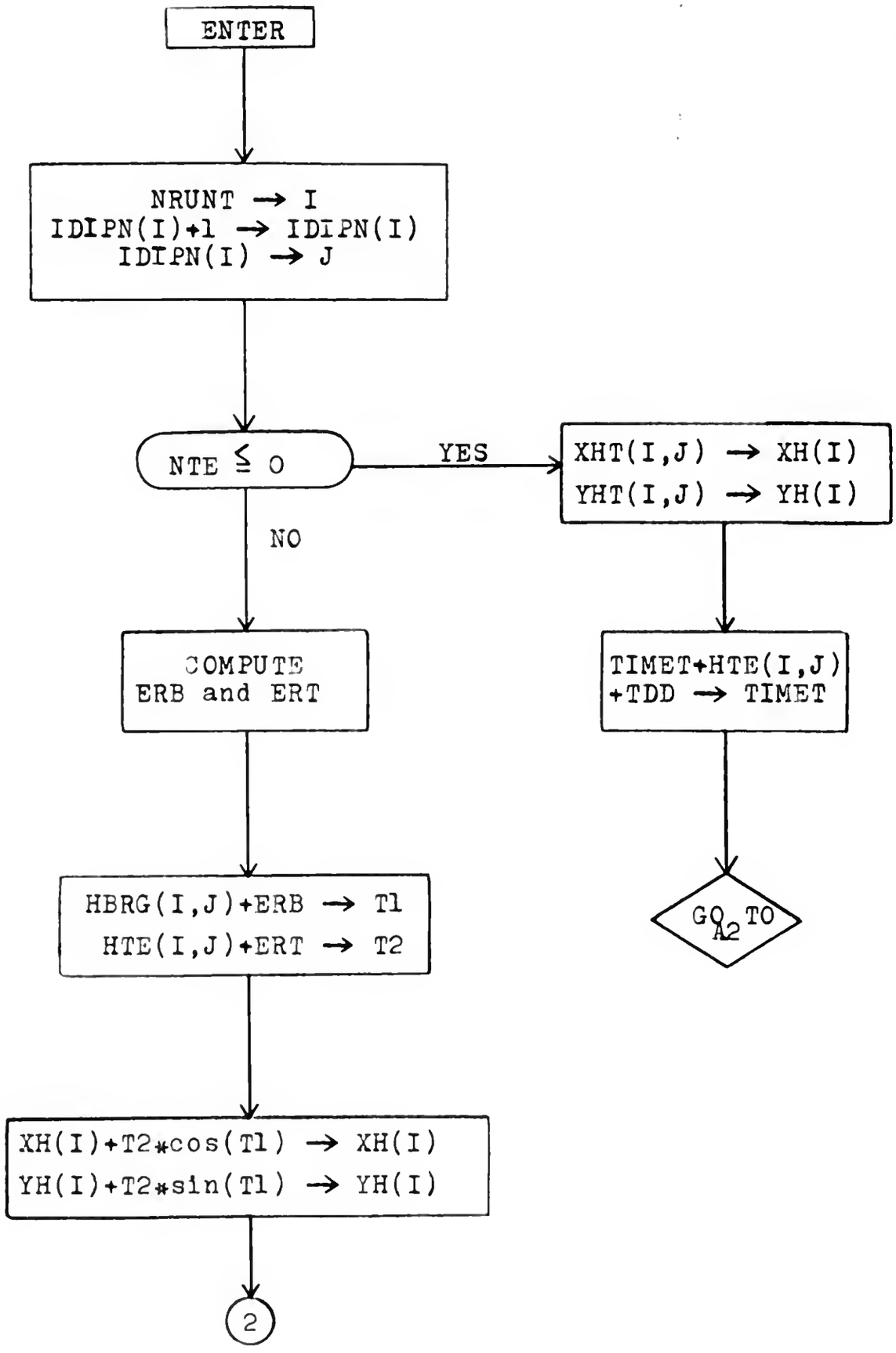
(2) Stores the next detection event for the helicopter which has maneuvered. The succeeding detection event will occur at the time the maneuver is completed plus the time which elapses while the transducer is being lowered into the water. The latter time includes normal delay time required to transition from forward to hovering flight and is an input to the simulation.

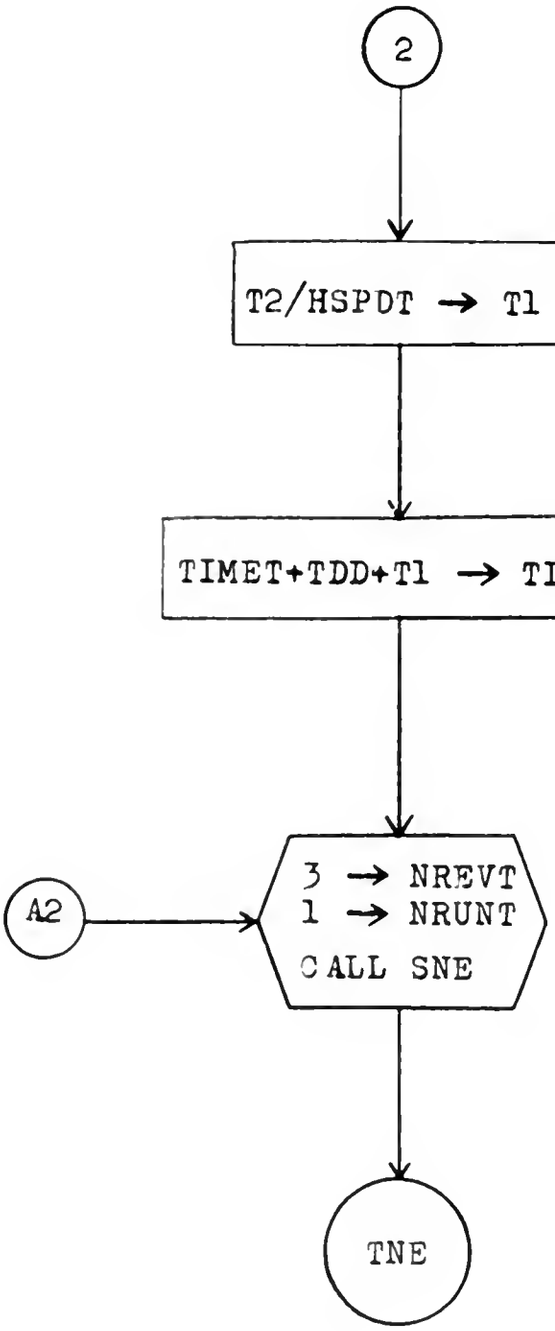
Subroutine HSM - Helicopter Maneuver Event (General)




2
COMPUTE
TIME ENROUTE
A2
ADD TIME ENROUTE
AND TIME TO
LOWER TRANSDUCER
TO GAME TIME
SNE
DETECTION EVENT
FOR THIS HELO TO
OCCUR AT TIME
COMPUTED ABOVE
TNE

Subroutine HSM – Helicopter Maneuver Event (Detailed)




2
T2/HSPDT → T1
TIMET+TDD+T1 → TIMET
A2
3 → NREVT
1 → NRUNT
CALL SNE
TNE

Subroutine HDE. Subroutine HDE (Detection Event) is called into action whenever a helicopter arrives at a new dip station. Using the information previously recorded by the submarine and helicopter maneuver events, the relative positions of helicopter and submarine are determined at the beginning of each sonar sweep. On the basis of this target range and helicopter detection range, a determination is made as to whether the submarine has been detected.

Additional functions performed by Subroutine HDE are as follows:

(1) Each time a helicopter dips the dip is recorded. When all helicopters have completed their assigned number of dips, or the submarine is detected, a flag is set which signals the end of the play.

(2) If the play terminates with a submarine detection, the elapsed time from when the helicopters arrived at datum until detection time, and the number of the detecting helicopter is recorded. This information is used by the main program in computing detection statistics.

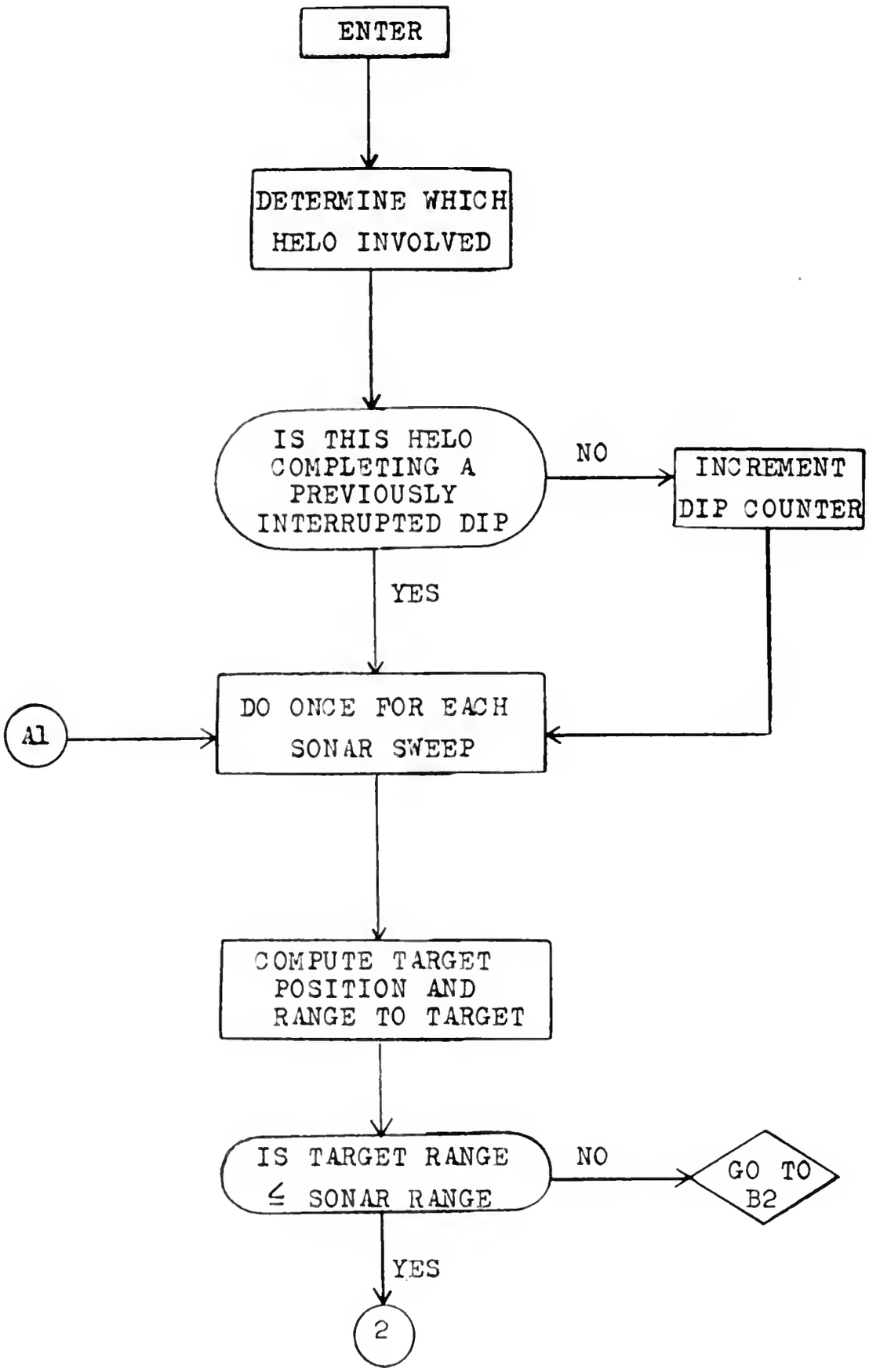
(3) A record of the time of the next intended submarine movement is maintained. If a submarine maneuver is to take place during any helicopter dip, the detection event is discontinued after the sonar sweep during which this maneuver is to occur. After the submarine maneuver has been executed the helicopter dip is resumed at the same game time at which the interruption occurred.

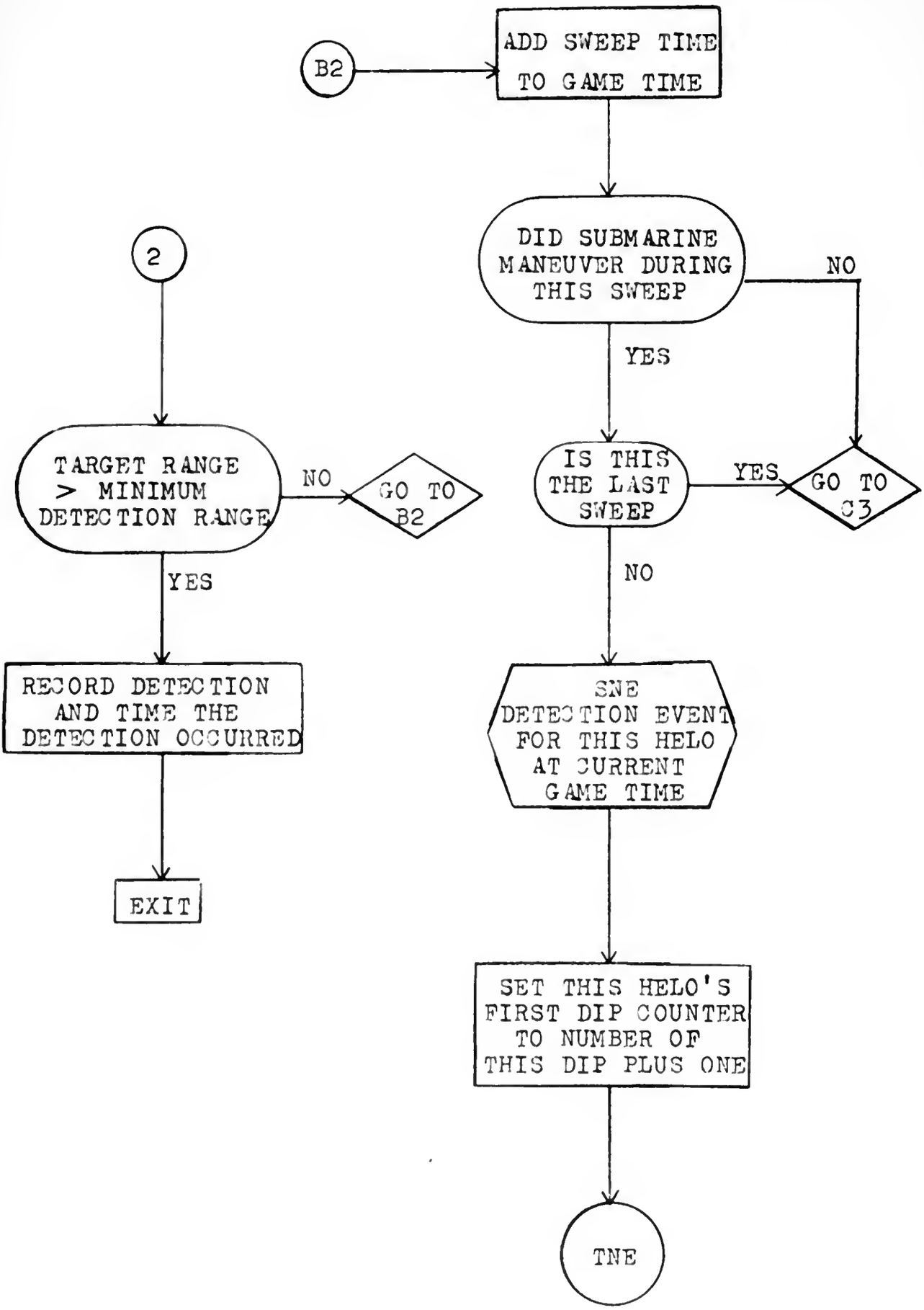
(4) If the probability of contacting a non-submarine target is greater than zero a random number is generated and compared with this probability. On the basis of this comparison and the maximum time to pursue non-submarine contacts input to the program, appropriate delays are computed and added to dip time.

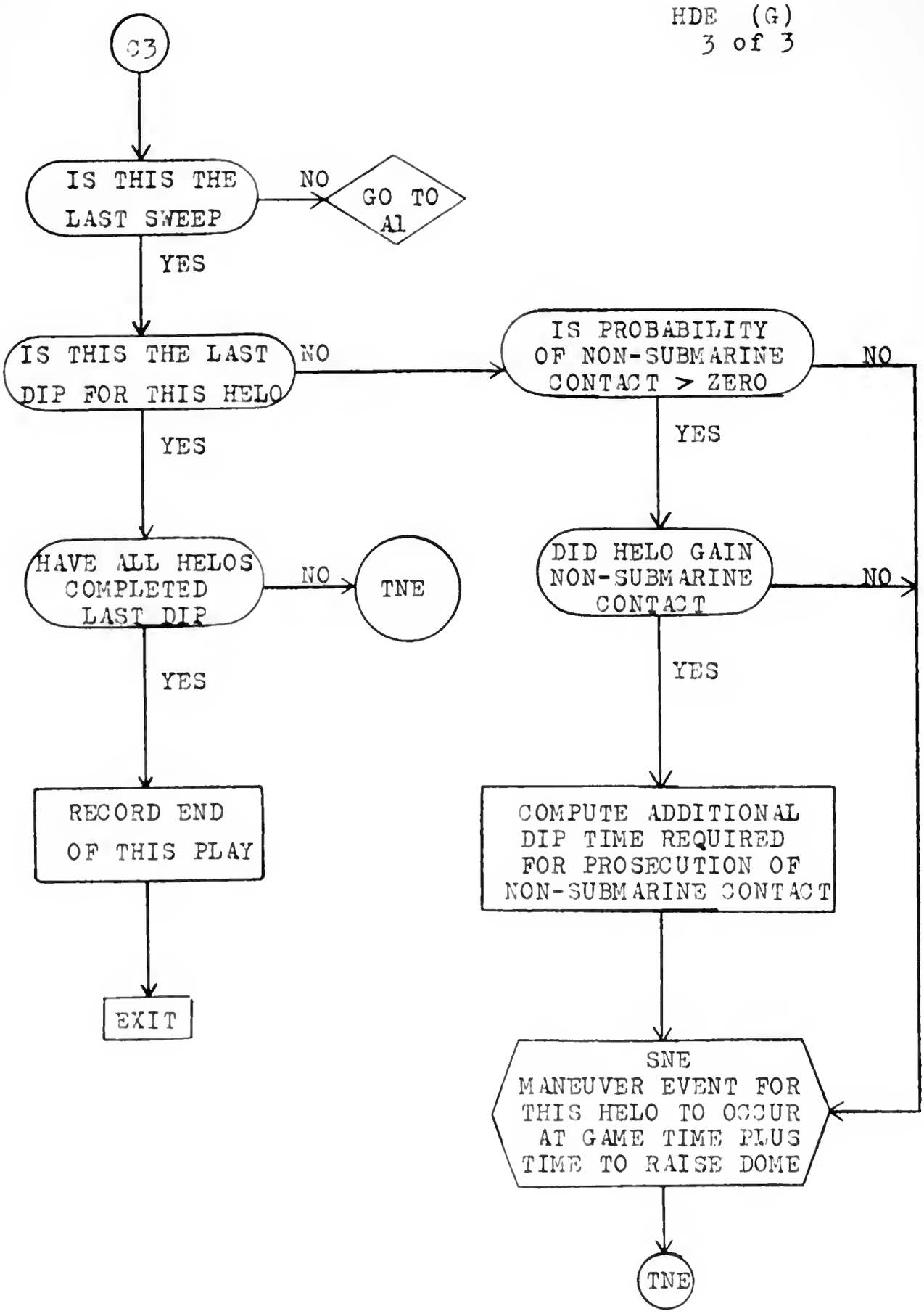
(5) On all but the last dip for each helicopter, a maneuver event is stored for the helicopter currently dipping. The time the maneuver is to take place is determined by adding dip time to the game time at which the dip commenced. Dip time is the sum of all sweep times, non-submarine contact delay time and the time required

for the helicopter to raise the transducer and make the transition from hovering to forward flight.

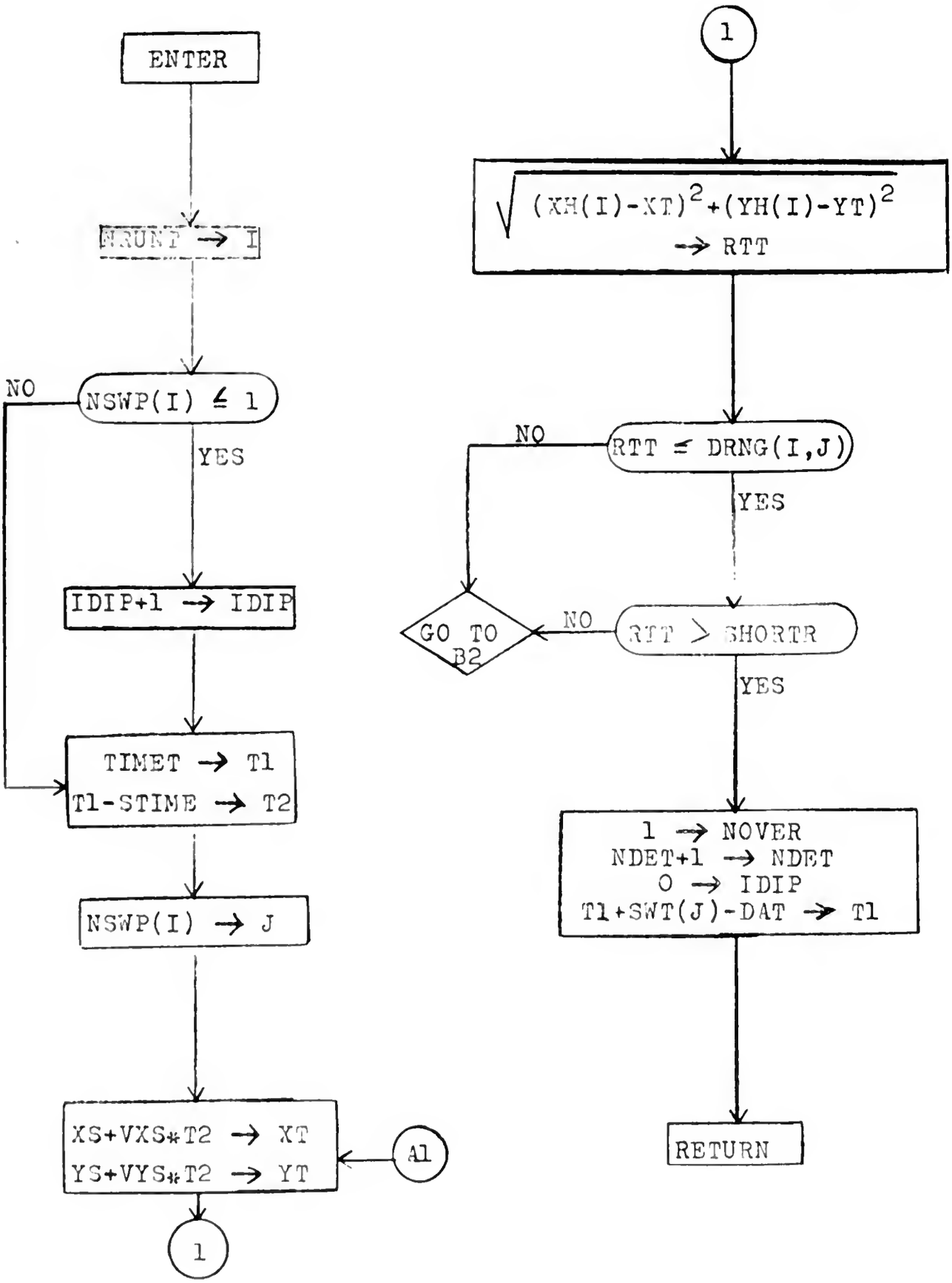
Subroutine HDE - Detection Event (General)

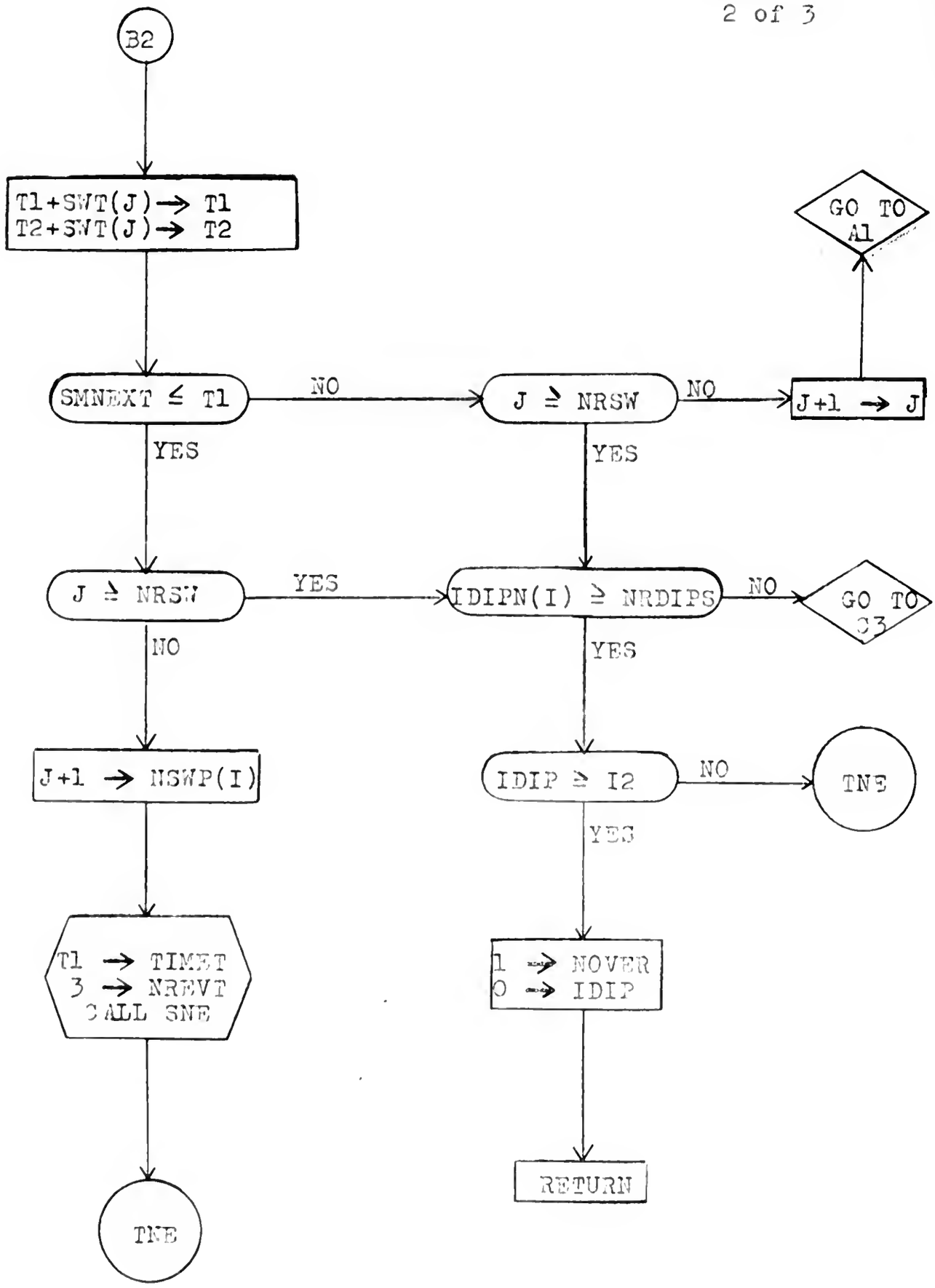


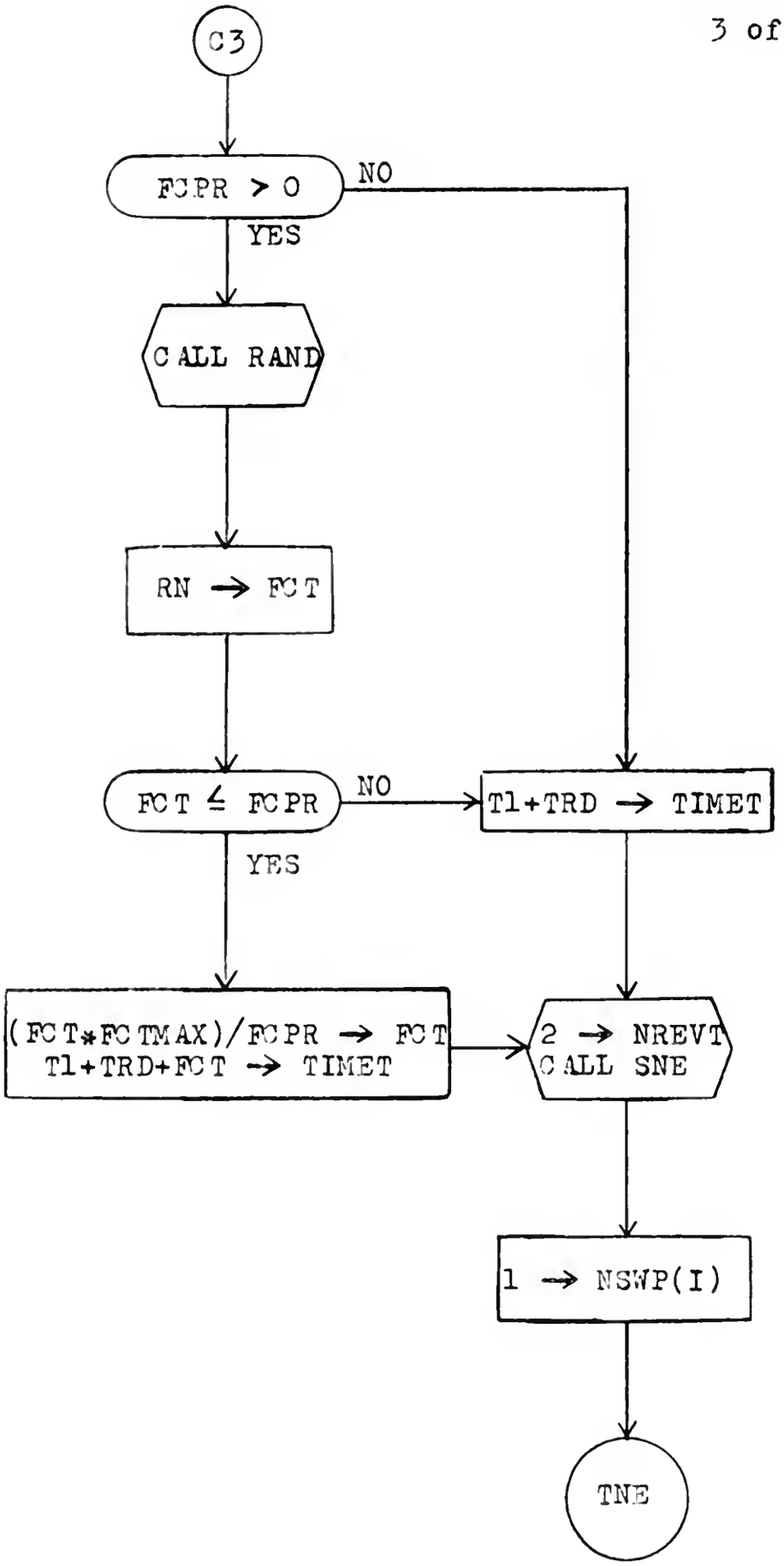

B2
ADD SWEEP TIME TO GAME TIME
DID SUBMARINE MANEUVER DURING THIS SWEEP
NO
YES
2
TARGET RANGE > MINIMUM DETECTION RANGE
NO
GO TO B2
YES
IS THIS THE LAST SWEEP
YES
GO TO C3
NO
RECORD DETECTION AND TIME THE DETECTION OCCURRED
SNE DETECTION EVENT FOR THIS HELO AT CURRENT GAME TIME
EXIT
SET THIS HELO'S FIRST DIP COUNTER TO NUMBER OF THIS DIP PLUS ONE
TNE


C3
IS THIS THE LAST SWEEP
NO
GO TO A1
YES
IS THIS THE LAST DIP FOR THIS HELO
NO
YES
HAVE ALL HELOS COMPLETED LAST DIP
NO
TNE
YES
RECORD END OF THIS PLAY
EXIT
IS PROBABILITY OF NON-SUBMARINE CONTACT > ZERO
NO
YES
DID HELO GAIN NON-SUBMARINE CONTACT
NO
YES
COMPUTE ADDITIONAL DIP TIME REQUIRED FOR PROSECUTION OF NON-SUBMARINE CONTACT
SNE
MANEUVER EVENT FOR THIS HELO TO OCCUR AT GAME TIME PLUS TIME TO RAISE DOME
TNE

Subroutine HDE - Detection Event (Detailed)




B2
T1+SWT(J) → T1
T2+SWT(J) → T2
SMNEXT ≤ T1
NO
J ≥ NRSW
NO
J+1 → J
GO TO A1
YES
YES
J ≥ NRSW
YES
IDIPN(I) ≥ NRDIPS
NO
GO TO C3
NO
YES
J+1 → NSWP(I)
IDIP ≥ I2
NO
TNE
YES
T1 → TIMET
3 → NREVT
CALL SNE
1 → NOVER
0 → IDIP
TNE
RETURN


C3
FCPR > 0
NO
YES
CALL RAND
RN → FCT
FCT ≤ FCPR
NO
T1+TRD → TIMET
YES
(FCT*FCTMAX)/FCPR → FCT
T1+TRD+FCT → TIMET
2 → NREVT
CALL SNE
1 → NSWP(I)
TNE

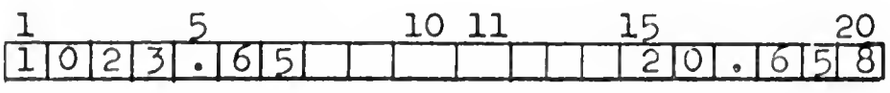
Subroutine PRINT. This subroutine comprises the Input-Output section of the program. It is called at the beginning of each run and at the completion of the last run. Input data cards are arranged in groups headed by a card having a Hollereth field name punched in the first six columns. This name signals the information to be read from the succeeding cards. This system allows the user to change any number of input quantities between runs without preparing data cards for the complete set of inputs. After the inputs for each run have been read into the machine, the input data are printed out and output data headings are printed.

Input data and rules for data card preparation. Three types of fields are used; External Fixed Point (F), Integer (I), and (A). All (F) fields are F10.5 and when this field is specified data may be punched using either one of two methods: (1) without the decimal point, requiring that the last digit prior to the decimal point be punched in the fifth column of the field, or (2) with the decimal point, in which case the number may fall any place within the field.

EXAMPLE: To read in the numbers 1023.65 and 20.658 using a 2F10.5 field:

(1) Without the decimal point:

(2) With the decimal point:



When an (I) field is specified the last digit of the number must fall in the last column of the field. The (A) field is used for

reading in literal characters and the letters or numbers to be read may fall any place within the six columns of the field.

All arrays within a group are read in an alternating sequence, i.e., If SWT (I) and PD(I) are to be read from a 6F10.5 field, I = 1,4 the following order is used.

Card	Columns	Name
1	1-10	SWT(1)
	11-20	PD(1)
	21-30	SWT(2)
	31-40	PD(2)
	41-50	SWT(3)
	51-60	PD(3)
2	1-10	SWT(4)
	11-20	PD(4)

Two dimensional arrays (Array (I,J)) are listed on the card in order of increasing J and a new card is started for each increase in I. If more than one array is read in a group, the names are alternated as for one dimensional tables. In the following instructions only the first elements are shown; succeeding elements will be in the order shown.

Data Card Format

GROUP I

Card	Columns	Field	Name	Remarks
1	1-6	A6	*DATE	
	15-16	I10	NDAY	Current date.
	25-26		MONTH	Number of current month.
	35-36		NYEAR	Last two digits of current year.

This Group must be the first card and must appear only once in a series of runs.

*Literal characters.

GROUP II

Card	Columns	Field	Name	Remarks
1	1-5	A6	*NRSAM	
	7-16	I10	NRSAM	Number of samples.
	26		KDET	Always equal to one when included. Signals that detection statistics are to be printed at end of run. (See output data)
	27-36		IRNC	
2 and additional cards if required	1-6	12I6	NSSIZE(1)	Number of entries must agree with NRSAM.
	7-12		NSSIZE(2)	
	.		.	
	.		.	
	.		.	

GROUP III

Card	Columns	Field	Name	Remarks
1	1-6	A6	*NHISPD	Submarine speed and depth limits.
	7-16	I10	NHISPD	Knots
	17-26		LOSPD	Knots
	27-36		MAXSD	Feet
	37-46		MINSD	Feet

This group is normally not included if submarine maneuvers are predetermined. However, it is legal to include both groups III and V in the same set of data. The data to be used for any subsequent run is then determined by the value of NRANSS.

GROUP IV

Card	Columns	Field	Name	Remarks
1	1-6	A6	*NRANSS	
	16	I6	NRANSS	Equals one when included.
	22		NRANC	Equals zero if random number is to be added to each submarine course, one otherwise.

This group need not be included unless submarine maneuvers are predetermined. The one exception is if both Groups III and V have been read on the same run. In this case NRANSS and NRANC must be read in each time the submarine mode of operation is to be changed. A zero in column 16 will result in random submarine maneuvers.

GROUP V

Card	Columns	Field	Name	Remarks
1	1-5	A6	*NRMAN	
	7-16	I10	NRMAN	Number of submarine maneuvers.
2	1-10	4F10.5	SMTIME(1)	Minutes (Time the maneuver is to take place) SMTIME(1) must always be 0.0.
	11-20		DEPTH(1)	Depth. (feet)
	21-30		SCUS(1)	Course. (degrees)
	31-40		SSPD(1)	Speed. (knots)
3	1-10		SMTIME(2)	
and as required	⋮		⋮	

GROUP VI

Card	Columns	Field	Name	Remarks
1	1-4	A6	*NRSW	
	7-16	I10	NRSW	Number of sonar sweeps
2	1-10	6F10.5	SWT(1)	Minutes to complete sweep.
	11-20		PD(1)	Transducer depth. (feet)
	21-30		SWT(2)	
	31-40		PD(2)	
	41-50		SWT(3)	
	51-60		PD(3)	

GROUP VII

Card	Columns	Field	Name	Remarks
1	1-3	A6	*TDD	
2	1-10	2F10.5	TDD	Minutes to lower transducer.
	11-20		TRD	Minutes to raise transducer.
	21-30		FCPR	$0 \leq FCPR \leq 1.0$
	31-40		FCTMAX	Maximum time to classify non-submarine targets. (Minutes).

GROUP VIII

Card	Columns	Field	Name	Remarks
1	1-4	A6	*NRHS	
	7-16	I10	NRHS	Number of helicopters.
	17-26		NRDIPS	Number of dips for each helicopter.
	27-36		HSPD	Helicopter speed (knots).

2 and additional cards as needed.	1-10	6F10.5	HBRG(1,1)	Bearing of first helicopter's first dip station. (Degrees relative to 000).
	11-20		HTE(1,1)	Jump distance (Yards)
	21-30		HBRG(1,2)	Degrees relative to HBRG(1,1)
	31-40		HTE(1,2)	
	41-50		HBRG(1,3)	Degrees relative to HBRG(1,2)
	⋮		⋮	
New Card	1-10	6F10,5	HBRG(2,1)	
	11-20		HTE(2,1)	
	⋮		⋮	

GROUP IX

Card	Columns	Field	Name	Remarks
1	1-4	A6	*XDAT	
	7-16	I10	XDAT	X and Y coordinates
	17-26		YDAT	of datum.

This group may be omitted in which case the datum coordinates will be (0,0).

GROUP X

Card	Columns	Field	Name	Remarks
1	1-5	A6	*SONAR	
	7-16	I10	NS	Sea State
2	1-10	6F10.5	FOMMU	Average sonar figure of merit. (Decibels)
	11-20		FOMVAR	Variance of FOM. (db)
	21-30		DL	Layer depth. (Feet)
	31-40		T	Temperature in the layer. (^{o}Fahrenheit)
	41-50		F	Sonar frequency. (kc.)
	51-60		TS	Target strength. (db.)

GROUP XI

Card	Columns	Field	Name	Remarks
1	1-3	A6	*DAT	
2	1-10	F10.5	DAT	Time late.(minutes)
	11-16	A6	NAME	Alpha-numeric characters (Name of search plan)

This group must appear in every run and must not be placed ahead of Groups I or II.

GROUP XII

Card	Columns	Field	Name	Remarks
1	1-2	A6	*GO	Signals program to print out inputs and commence new run. Must appear at the end of each set of data.

GROUP XIII

Card	Columns	Field	Name	Remarks
1	1-5	A6	*FINIS	Signals program to compute detection statistics at the end of the last run and stop the machine. Need not appear if detection statistics are not desired.

Output data. The output statistics reflect the writer's experience in ASW Helicopter squadrons. No assurance is given that they are suited to any other potential users requirements. It is relatively easy, however, to write the coding necessary to compute and print any desired output.

The following output data is printed at the completion of every sample:

(1) Sample size.

(2) Number of detections. This is the primary measure of effectiveness associated with this simulation.

(3) Percentage of plays resulting in detections. This is included principally as an aid in comparing samples of different size.

$$\text{Percentage} = \frac{\text{Number of detections}}{\text{Sample size}} \cdot 100 \qquad (6.5)$$

(4) Average time to detect when a detection occurs. The time to complete those plays which do not end with a submarine detection are not included in this

average. Time to detect is measured from the time the helicopters arrive at datum. A problem encountered in fleet ASW operations is when to abandon a search once begun. Average time to detect should give the user some indication of the time at which further search with a particular search plan is apt to be unproductive.

(5) Sample variance of time to detect.

$$S^2 = (1/(n-1)) \sum_{i=1}^{n} (X_i - \bar{X})^2 \qquad (6.6)$$

where: n = number of detections

X_i = time to detect for the i^{th} detection.

$\bar{X}$ = average time to detect.

S^2 = sample variance of time to detect.

(6) Minimum time from datum time to detection

(7) Maximum time from datum time to detection. Neither (6) nor (7) include plays which do not end in a submarine detection.

(8) Number of detections by each helicopter. In general considerable discretion must be used in drawing inferences from individual unit performance. Treated with due caution, however, some meaningful information may be gained from this knowledge. This particularly applies to asymmetrical or random station search plans.

Printing of the above information is predetermined by the computer coding and the user has no control over whether the data is printed or not. The succeeding statistics are computed and written out after the last run and at the completion of any run determined by the user.

(1) Average number of detections per sample. For example: If this is printed for the first time at the end of run two, and runs one and two each consist of two samples.

$$\bar{D} = (\tfrac{1}{4}) \sum_{i=1}^{4} D_i \qquad (6.7)$$

where: $\bar{D}$ = average number of detections per sample.

D_i = the number of detections for the i^{th} sample.

(2) Sample variance of number of detections per sample. Referring to the above example:

$$S_D^2 = (1/3)\sum_{i=1}^{4} (D_i - \bar{D})^2 \qquad (6.8)$$

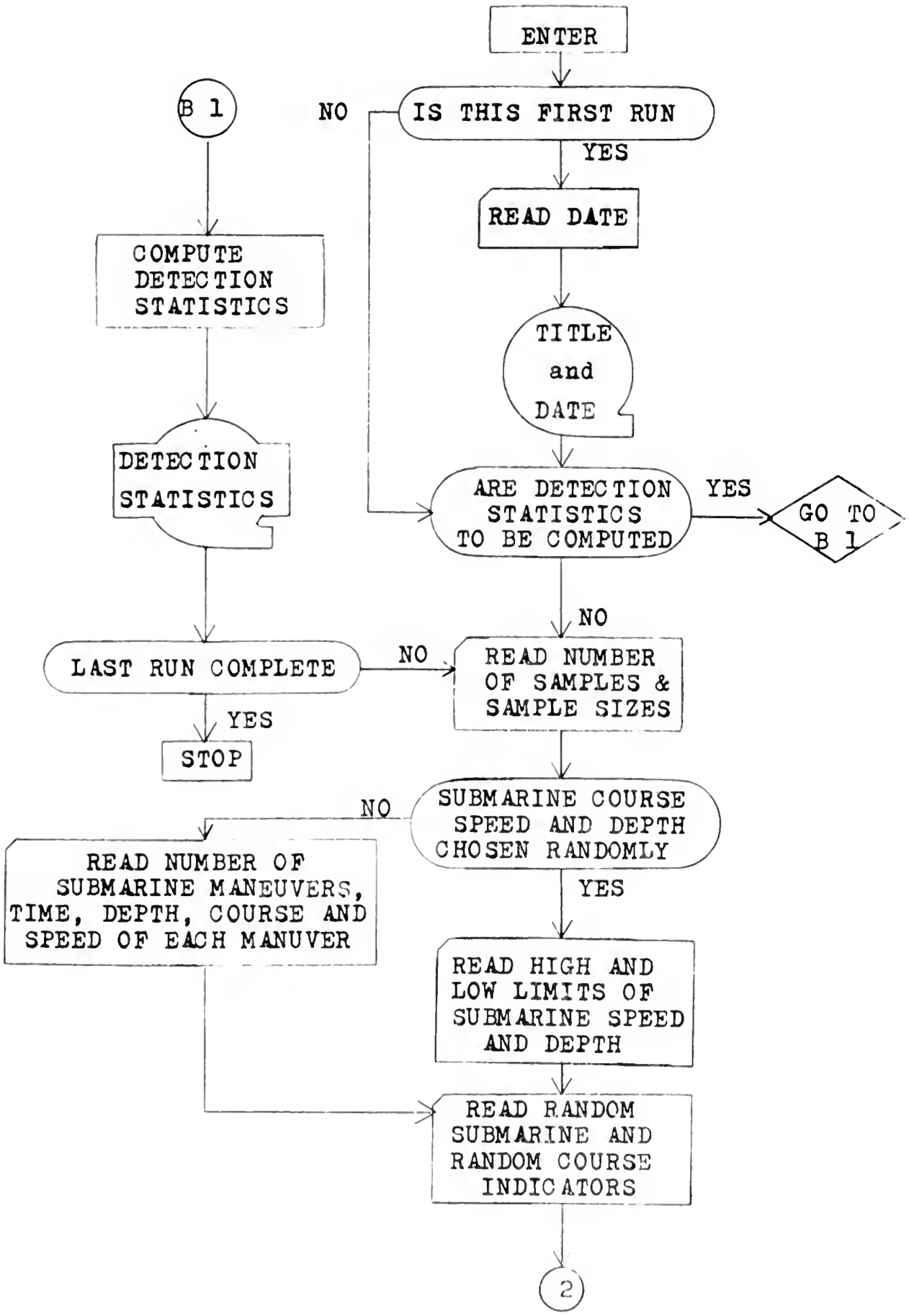
where: S_D = sample variance and D_i and $\bar{D}$ are as defined above.

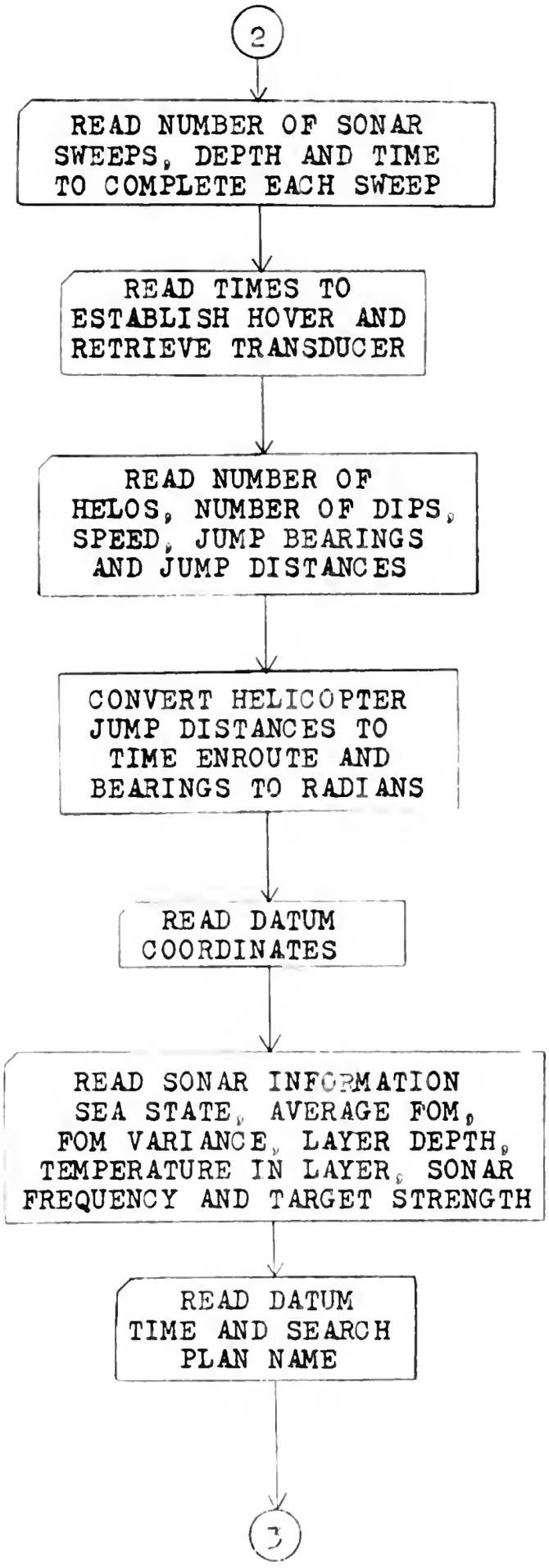
(3) Maximum number of detections taken over all samples.

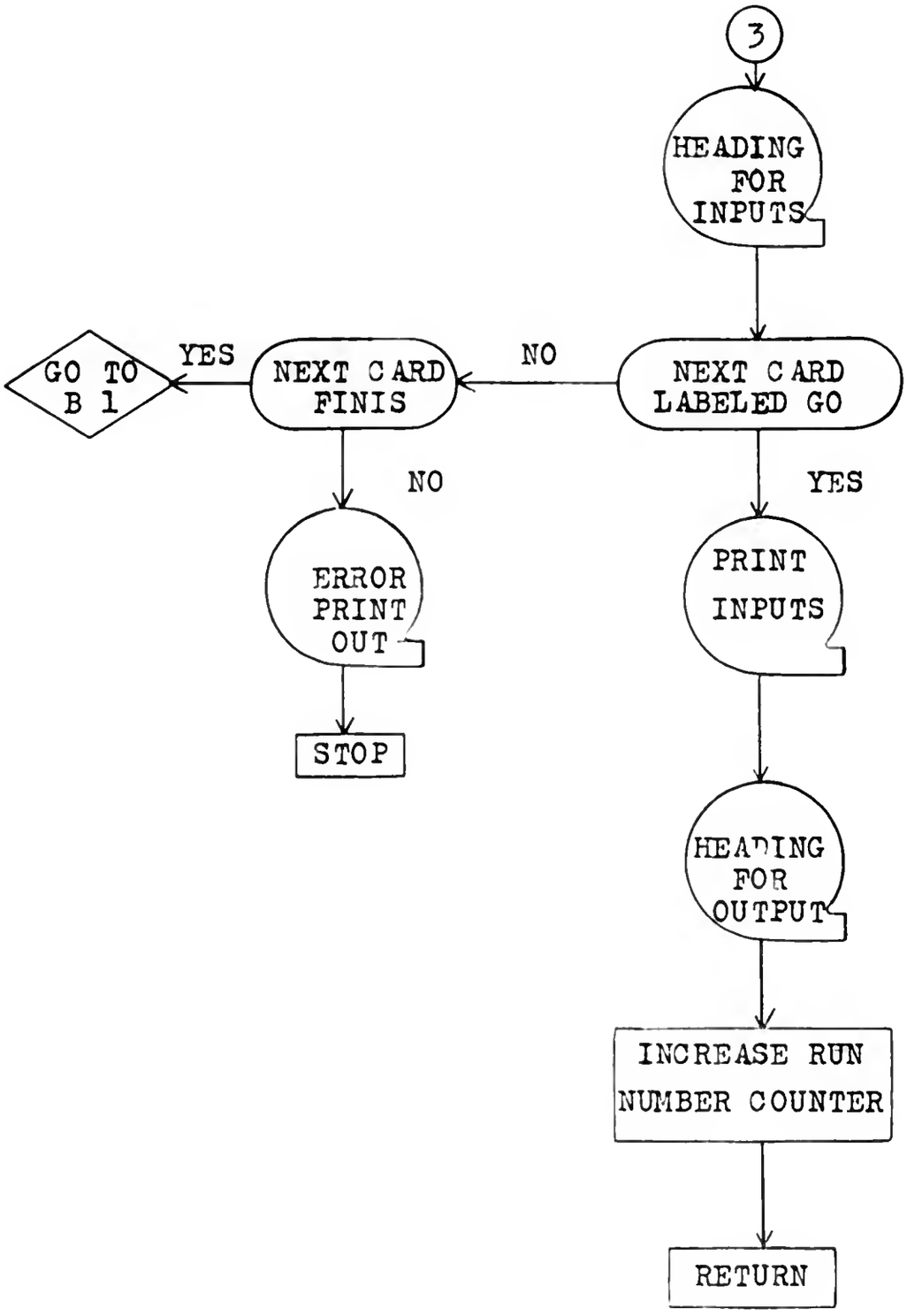
(4) Minimum number of detections taken over all samples.

The above quantities have meaning only if all samples are of the same size. It is therefore, anticipated that the user would want to compute and print this data whenever sample size is changed. Note that when the data is called for on any run, the average is taken over all runs since the last time these quantities were printed.

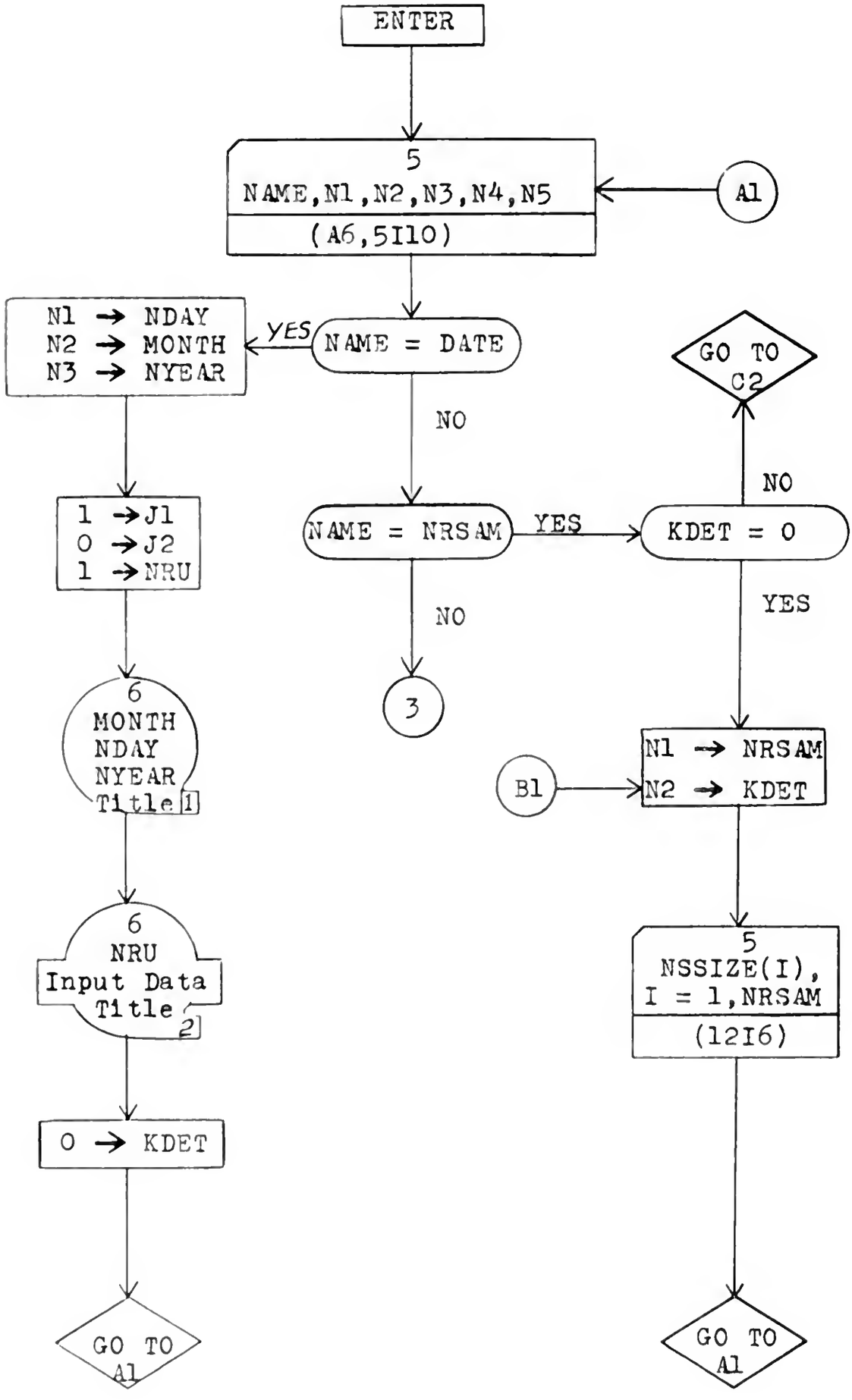
Subroutine PRINT General

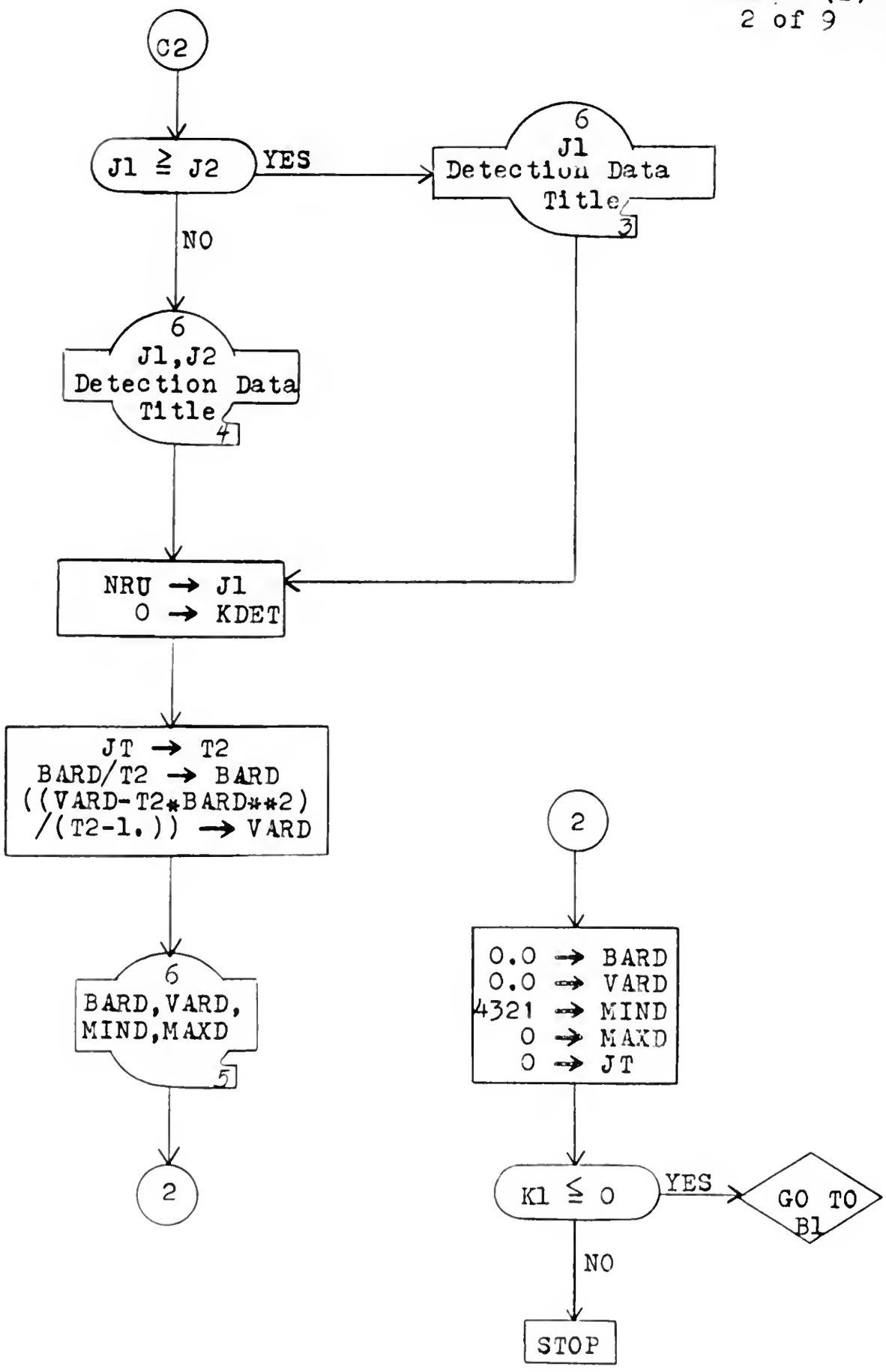


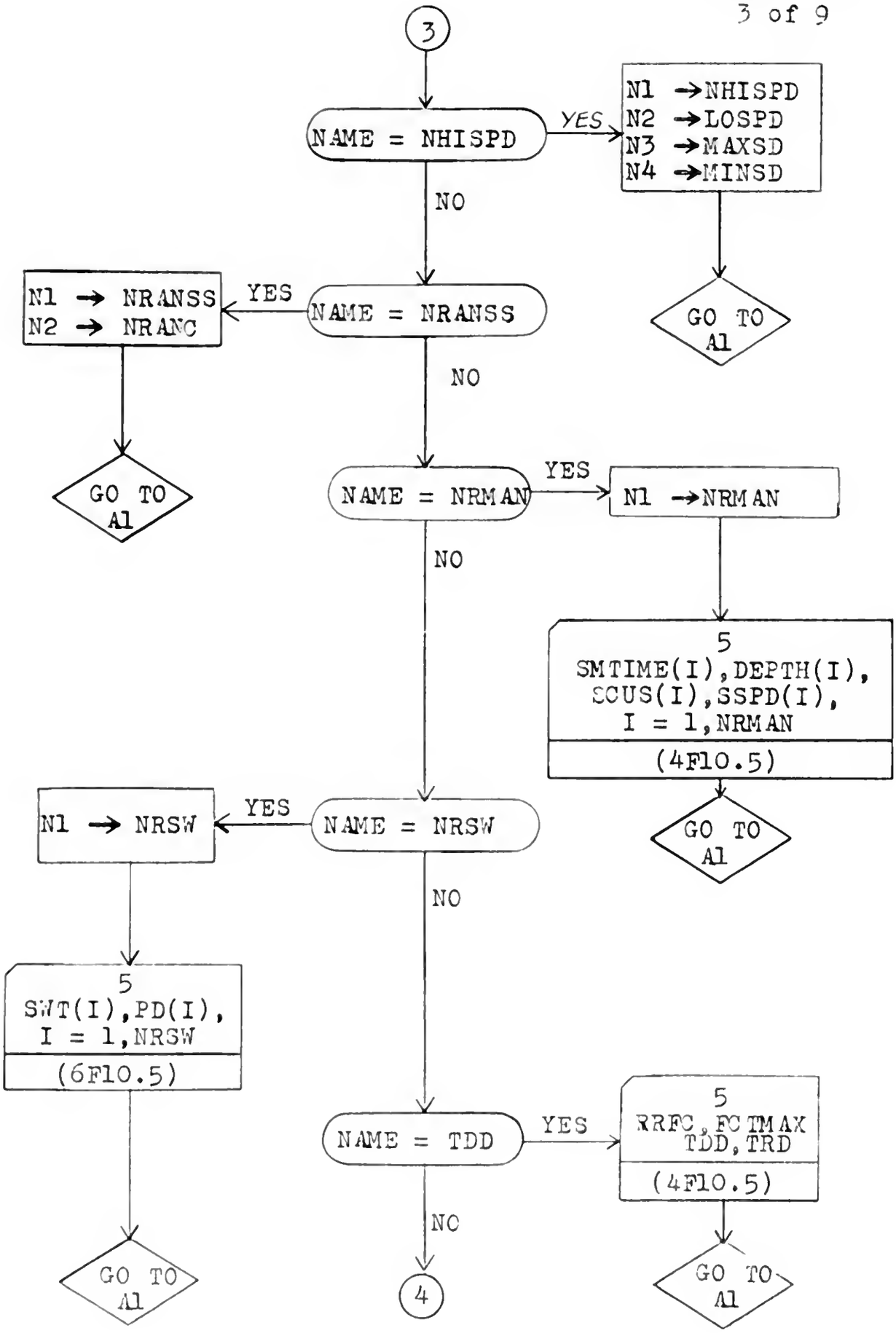

2
READ NUMBER OF SONAR SWEEPS, DEPTH AND TIME TO COMPLETE EACH SWEEP
READ TIMES TO ESTABLISH HOVER AND RETRIEVE TRANSDUCER
READ NUMBER OF HELOS, NUMBER OF DIPS, SPEED, JUMP BEARINGS AND JUMP DISTANCES
CONVERT HELICOPTER JUMP DISTANCES TO TIME ENROUTE AND BEARINGS TO RADIANS
READ DATUM COORDINATES
READ SONAR INFORMATION SEA STATE, AVERAGE FOM, FOM VARIANCE, LAYER DEPTH, TEMPERATURE IN LAYER, SONAR FREQUENCY AND TARGET STRENGTH
READ DATUM TIME AND SEARCH PLAN NAME
3

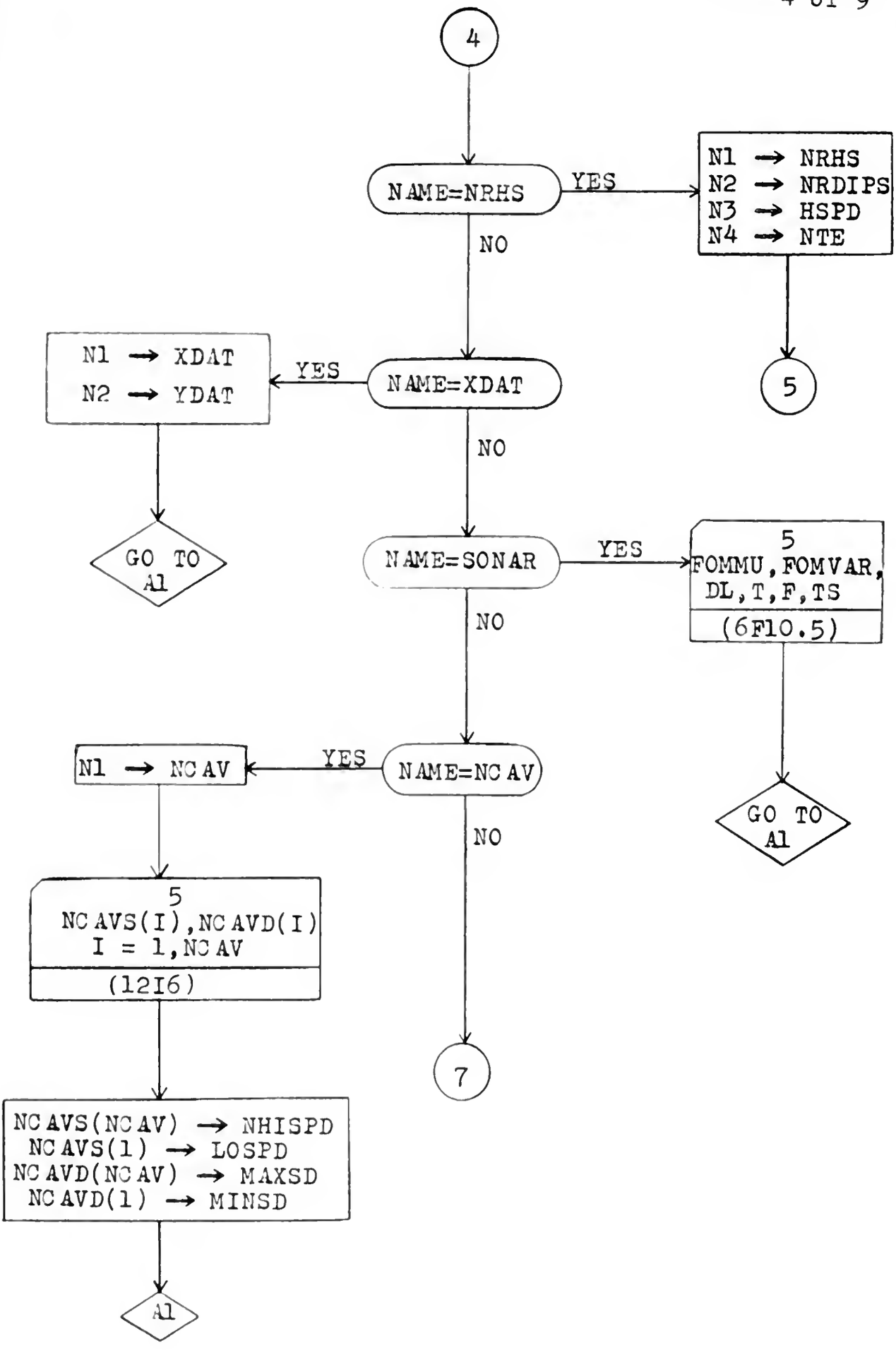

3
HEADING FOR INPUTS
NEXT CARD LABELED GO
NO
NEXT CARD FINIS
YES
GO TO B 1
NO
ERROR PRINT OUT
STOP
YES
PRINT INPUTS
HEADING FOR OUTPUT
INCREASE RUN NUMBER COUNTER
RETURN

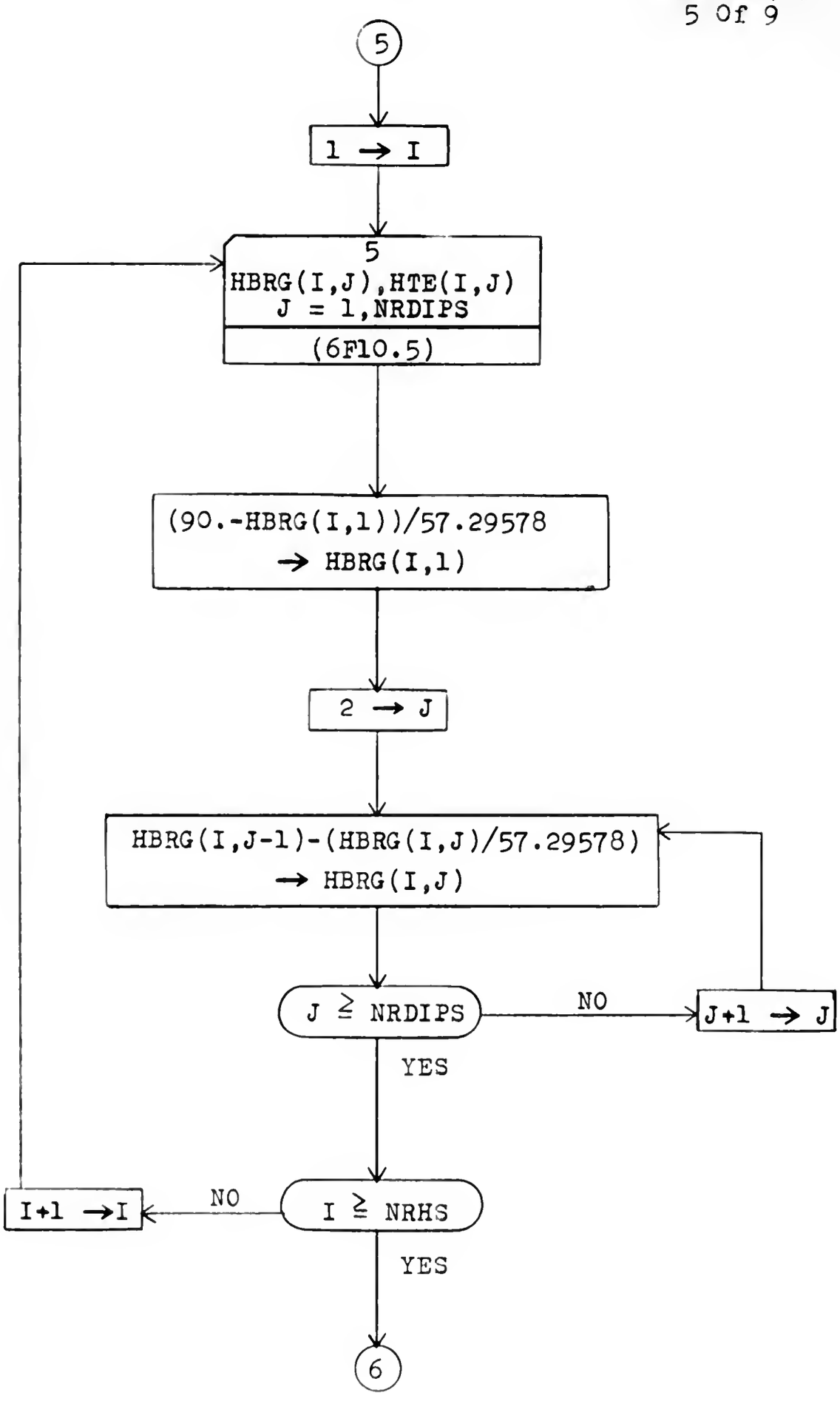
Subroutine PRINT (Detailed)

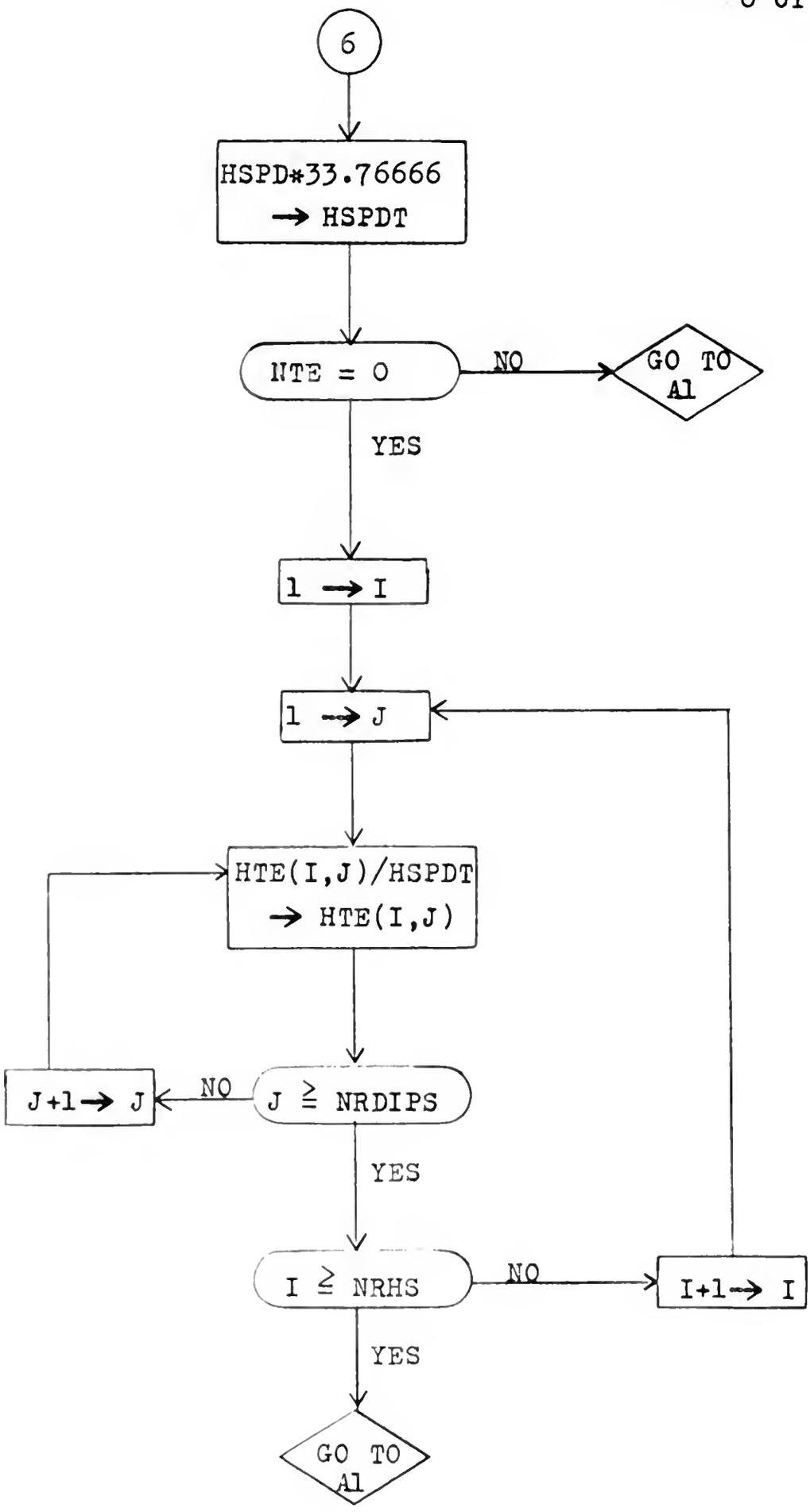


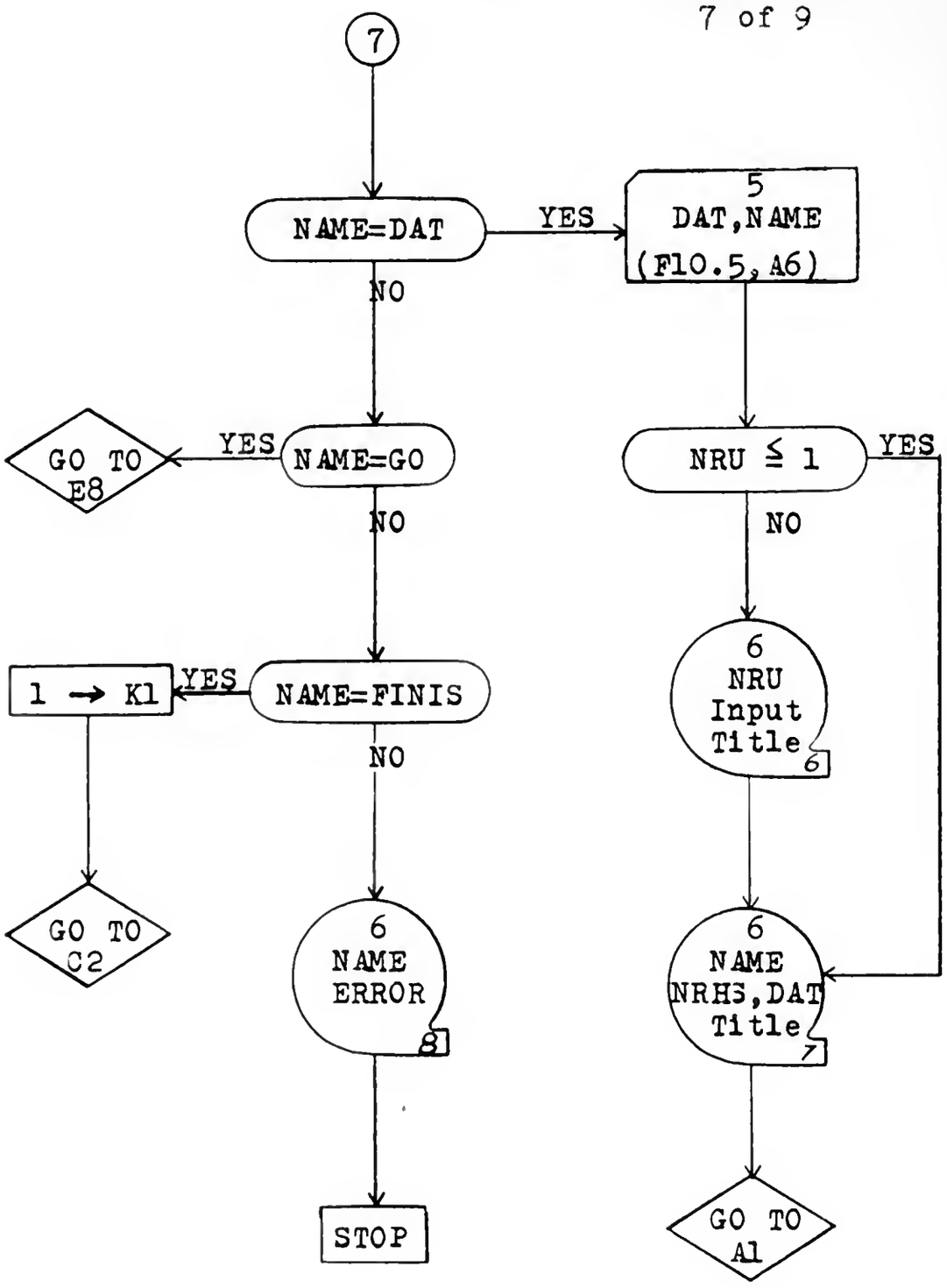

C2
J1 ≧ J2
YES
NO
6
J1
Detection Data
Title
3
6
J1,J2
Detection Data
Title
4
NRU → J1
0 → KDET
JT → T2
BARD/T2 → BARD
((VARD-T2*BARD**2)
/(T2-1.)) → VARD
6
BARD,VARD,
MIND,MAXD
5
2
2
0.0 → BARD
0.0 → VARD
4321 → MIND
0 → MAXD
0 → JT
K1 ≦ 0
YES
GO TO
B1
NO
STOP

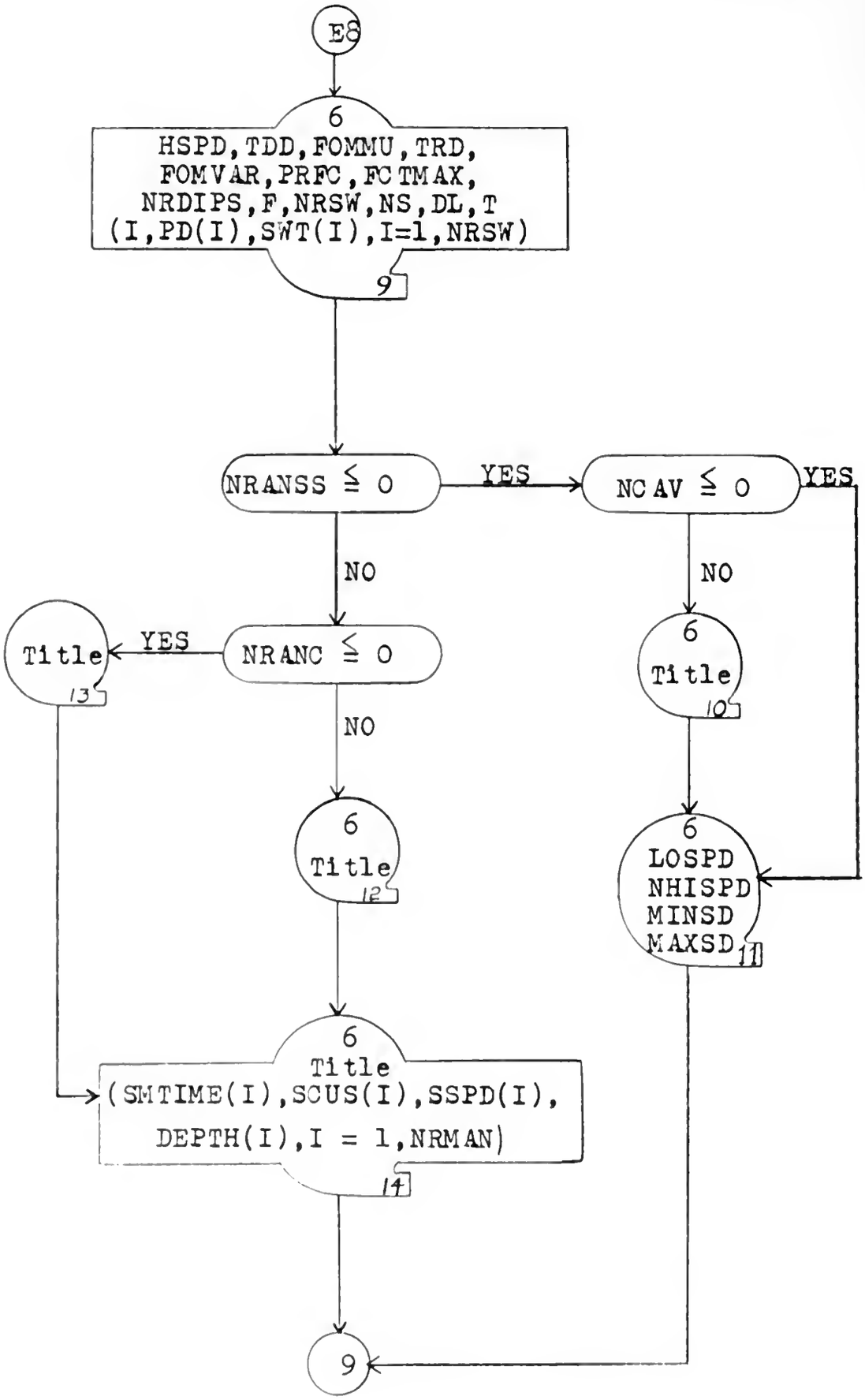

3
NAME = NHISPD
YES
N1 →NHISPD
N2 →LOSPD
N3 →MAXSD
N4 →MINSD
NO
GO TO A1
NAME = NRANSS
YES
N1 → NRANSS
N2 → NRANC
NO
GO TO A1
NAME = NRMAN
YES
N1 →NRMAN
NO
5
SMTIME(I),DEPTH(I),
SCUS(I),SSPD(I),
I = 1,NRMAN
(4F10.5)
GO TO A1
NAME = NRSW
YES
N1 → NRSW
NO
5
SWT(I),PD(I),
I = 1,NRSW
(6F10.5)
GO TO A1
NAME = TDD
YES
5
RRFC, FC TMAX
TDD, TRD
(4F10.5)
GO TO A1
NO
4

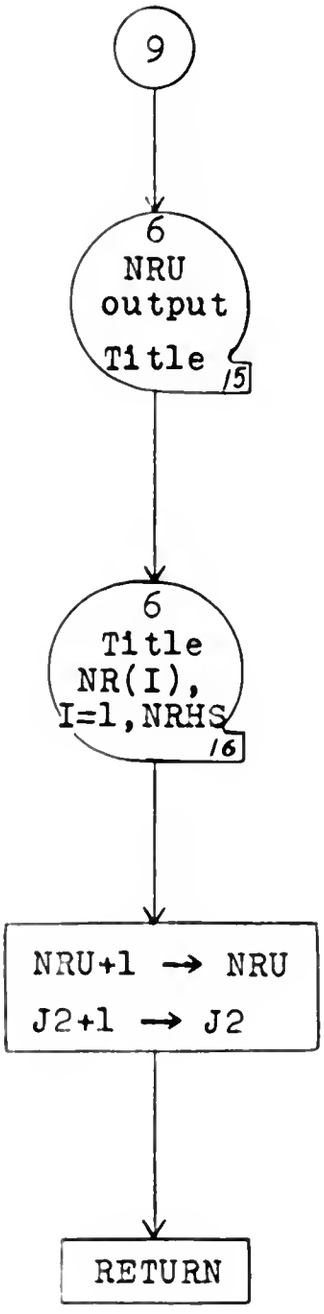

4
NAME=NRHS
YES
N1 → NRHS
N2 → NRDIPS
N3 → HSPD
N4 → NTE
NO
5
N1 → XDAT
N2 → YDAT
YES
NAME=XDAT
NO
GO TO
A1
NAME=SONAR
YES
5
FOMMU, FOMVAR,
DL, T, F, TS
(6F10.5)
NO
GO TO
A1
N1 → NCAV
YES
NAME=NCAV
NO
5
NCAVS(I), NCAVD(I)
I = 1, NCAV
(12I6)
7
NCAVS(NCAV) → NHISPD
NCAVS(1) → LOSPD
NCAVD(NCAV) → MAXSD
NCAVD(1) → MINSD
A1


5
1 → I
5
HBRG(I,J),HTE(I,J)
J = 1,NRDIPS
(6F10.5)
(90.-HBRG(I,1))/57.29578
→ HBRG(I,1)
2 → J
HBRG(I,J-1)-(HBRG(I,J)/57.29578)
→ HBRG(I,J)
J ≥ NRDIPS
NO
J+1 → J
YES
I ≥ NRHS
NO
I+1 →I
YES
6


6
HSPD*33.76666
→ HSPDT
NTE = 0
NO
GO TO
A1
YES
1 → I
1 → J
HTE(I,J)/HSPDT
→ HTE(I,J)
J+1 → J
NO
J ≧ NRDIPS
YES
I ≧ NRHS
NO
I+1 → I
YES
GO TO
A1

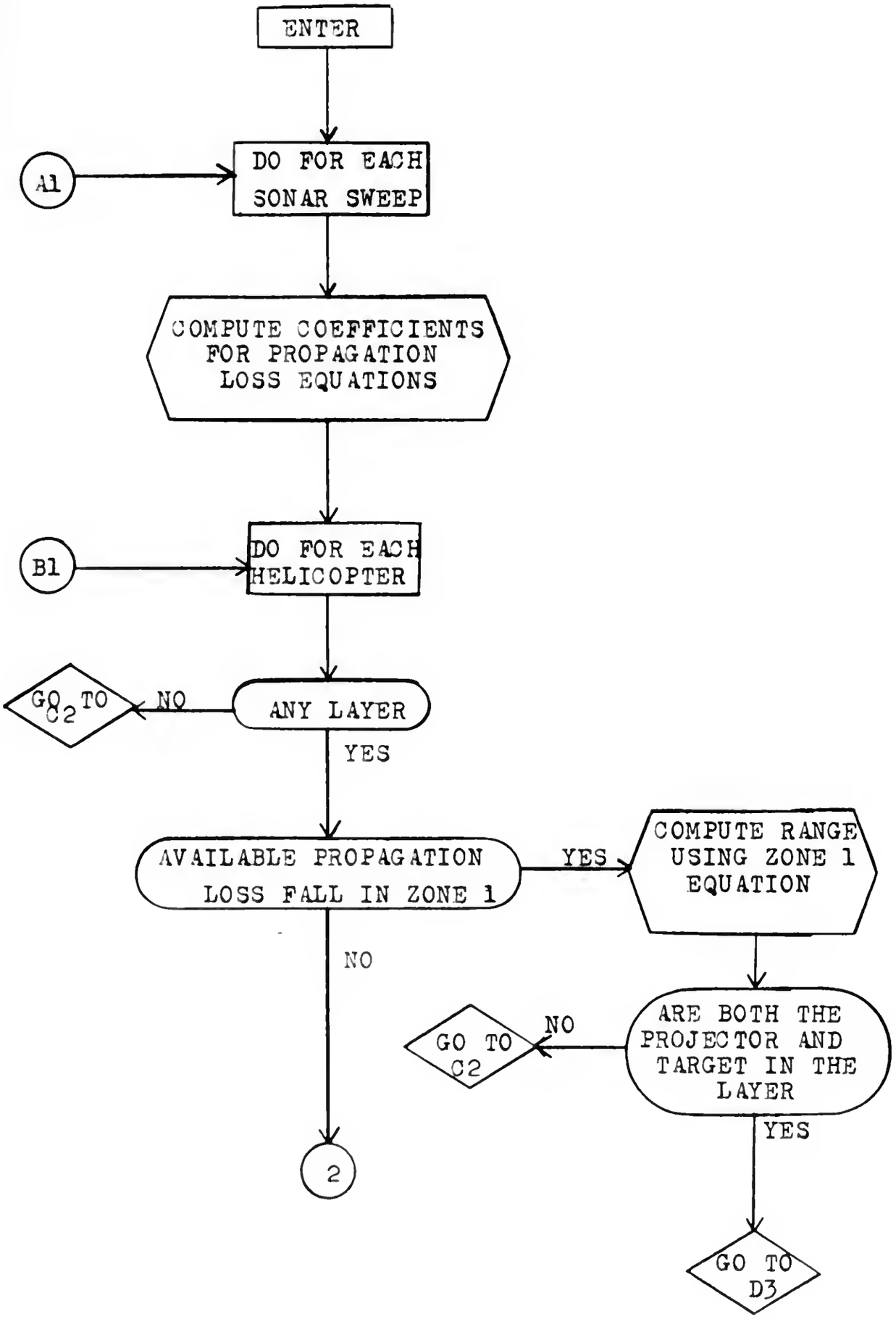

7
NAME=DAT
YES
5
DAT,NAME
(F10.5,A6)
NO
GO TO
E8
YES
NAME=GO
NRU ≤ 1
YES
NO
NO
1 → K1
YES
NAME=FINIS
6
NRU
Input
Title
6
NO
GO TO
C2
6
NAME
ERROR
8
6
NAME
NRH3,DAT
Title
7
STOP
GO TO
A1

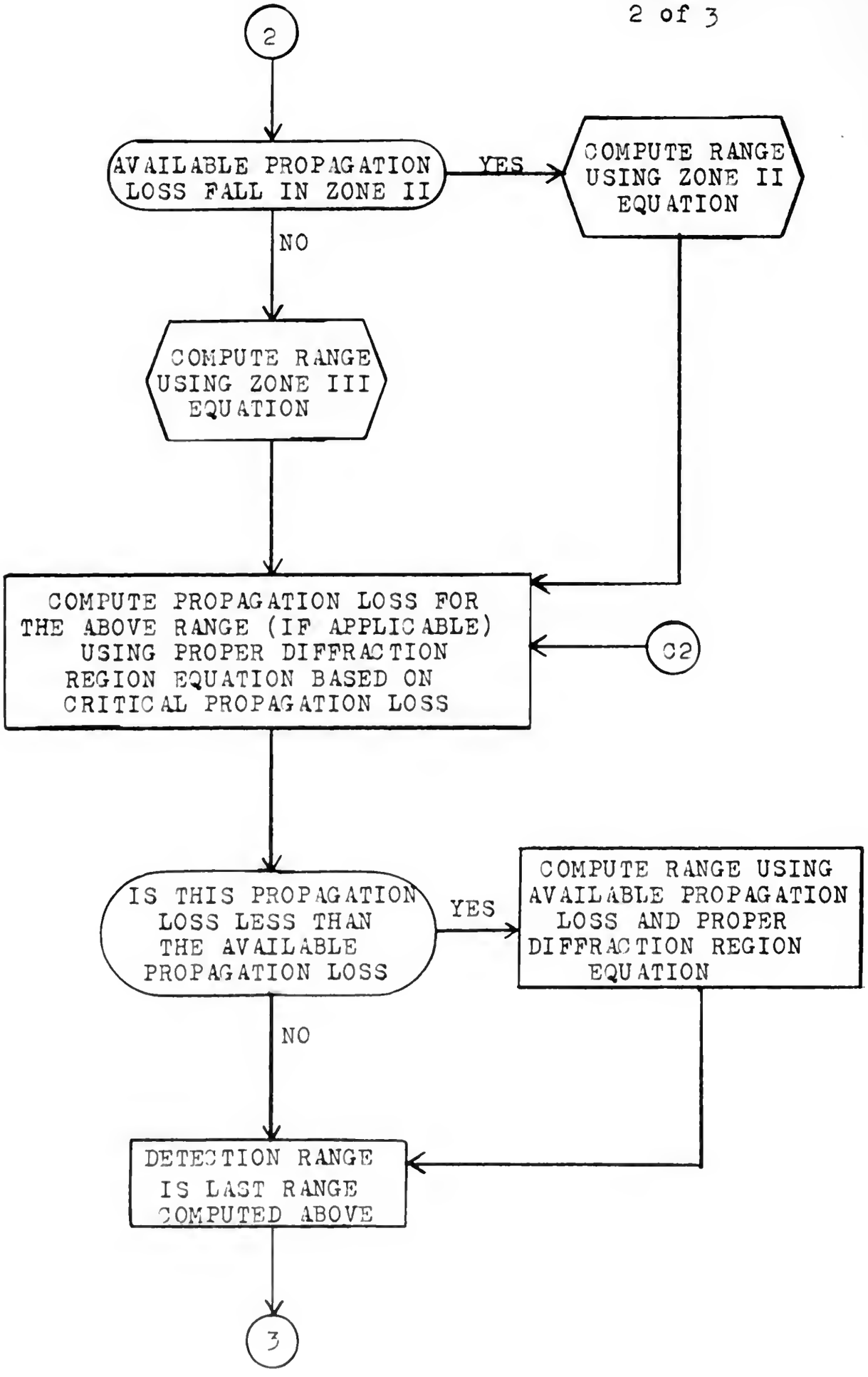

E8
6
HSPD,TDD,FOMMU,TRD,
FOMVAR,PRFC,FCTMAX,
NRDIPS,F,NRSW,NS,DL,T
(I,PD(I),SWT(I),I=1,NRSW)
9
NRANSS ≦ 0
YES
NC AV ≦ 0
YES
NO
NO
Title
13
YES
NRANC ≦ 0
6
Title
10
NO
6
Title
12
6
LOSPD
NHISPD
MINSD
MAXSD
11
6
Title
(SMTIME(I),SCUS(I),SSPD(I),
DEPTH(I),I = 1,NRMAN)
14
9


9
6
NRU
output
Title
15
6
Title
NR(I),
I=1,NRHS
16
NRU+1 → NRU
J2+1 → J2
RETURN

Subroutine COMPRG. This subroutine controls the computation of sonar detection ranges for each helicopter. The detection ranges are computed by an iterative process using the applicable zone or diffraction region equations. These regions are determined by the sonar conditions input to the program, submarine depth, transducer depth and the helicopter figure of merit.

COMPRG is a modification of a program written by R. L. Klinkner of the Applied Physics Laboratory, Johns Hopkins University. Subroutines SCR(KP) and SETCOF together with subroutine COMPRG accomplish the sonar detection range computations. The first two of these subroutines are used as they were written by Mr. Klinkner.


ENTER
A1
DO FOR EACH SONAR SWEEP
COMPUTE COEFFICIENTS FOR PROPAGATION LOSS EQUATIONS
B1
DO FOR EACH HELICOPTER
ANY LAYER
NO
GO TO C2
YES
AVAILABLE PROPAGATION LOSS FALL IN ZONE 1
YES
COMPUTE RANGE USING ZONE 1 EQUATION
NO
ARE BOTH THE PROJECTOR AND TARGET IN THE LAYER
NO
GO TO C2
YES
GO TO D3
2


2
AVAILABLE PROPAGATION LOSS FALL IN ZONE II
YES
COMPUTE RANGE USING ZONE II EQUATION
NO
COMPUTE RANGE USING ZONE III EQUATION
COMPUTE PROPAGATION LOSS FOR THE ABOVE RANGE (IF APPLICABLE) USING PROPER DIFFRACTION REGION EQUATION BASED ON CRITICAL PROPAGATION LOSS
C2
IS THIS PROPAGATION LOSS LESS THAN THE AVAILABLE PROPAGATION LOSS
YES
COMPUTE RANGE USING AVAILABLE PROPAGATION LOSS AND PROPER DIFFRACTION REGION EQUATION
NO
DETECTION RANGE IS LAST RANGE COMPUTED ABOVE
3

```
      SUBROUTINE TNE
C
C          *** DIMENSION AND COMMON STATEMENTS ***
C
    8 IF( NOVER) 9,9,14
    9 TIMET=TIME(2)
      NRUNT =NRUN(2)
      NREVT =NREV(2)
C
      DO 10 I   = 3,NTNE
      TIME(I-1) = TIME(I)
      NRUN(I-1) = NRUN(I)
      K = I
   10 NREV(I-1) = NREV(I)
      NTNE = NTNE - 1
C
      GO TO (11,12,13), NREVT
   11 CALL SME
      GO TO 8
   12 CALL HSM
      GO TO 8
   13 CALL HDE
      GO TO 8
   14 RETURN
      END
C
      SUBROUTINE SNE
C
C          *** DIMENSION AND COMMON STATEMENTS ***
C
C SEARCH EVENT STORE TABLE FOR PROPER LOCATION OF THIS EVENT.
      IT = NTNE
      NTNE =NTNE + 1
   14 IF(TIMET - TIME(IT)) 15,16,16
   15 TIME(IT+1) = TIME(IT)
      NRUN(IT+1) = NRUN(IT)
      NREV(IT+1) = NREV(IT)
      IT = IT - 1
      GO TO 14
   16 TIME(IT+1) = TIMET
      NRUN(IT +1) = NRUNT
      NREV(IT +1) = NREVT
      RETURN
      END
C
      SUBROUTINE HSM
C
C          *** DIMENSION AND COMMON STATEMENTS ***
C
      I = NRUNT
      IDIPN(I)=IDIPN(I)+1
  200 J = IDIPN(I)
      IF(NTE)201,201,260
  201 XH(I)=XHT(I,J)
      YH(I)=YHT(I,J)
      TIMET=TIMET+HTE(I,J)+TDD
      GO TO 270
C ROUTINE FOR COMPUTING BEARING AND TIMING ERRORS MAY BE
C   INSERTED HERE.
  260 T1=HBRG(I,J)+ERB
      T2         =HTE(I,J)+ERT
C  COMPUTE NEW X AND Y COORDINATES, HTE IS JUMP DISTANCE.
      XH(I) = XH(I) + T2*COSF(T1)
      YH(I) = YH(I) + T2*SINF(T1)
C
      T1 = T2/HSPDT
C  SET UP DETECTION EVENT FOR THIS HELICOPTER
      TIMET=TIMET +TDD + T1
  270 NREVT =3
      NRUNT = I
      CALL SNE
  280 RETURN
      END
```

```
      SUBROUTINE SME
C
C          *** CIMENSION AND COMMON STATEMENTS ***
C
      IF(KT-1) 100,100,108
  100 IF(NRANSS ) 101,101,102
C  DETERMINE SUBMARINE COURSE AND CONVERT DEGREES TO RADIANS
  101 RNC=360
      NXIN=RNC*RNG(1)
      T1=FLOATF(NXIN) /57.29578
C  DETERMINE SUBMARINE SPEED AND CONVERT TO YDS/MINUTE
    2 IF(NCAV) 3,3,4
    3 RNC=NHISPD-LOSPD+1
      NXIN=RNC*RNG(1)
      T2=FLOATF(LOSPD+NXIN)*33.766666
C  DETERMINE DEPTH
      RNC=MAXSD-MINSD+1
      NXIN=RNC*RNG(1)
      DT= MINSD+NXIN
      GO TO 10
C
    4 RNC=NCAVS(NCAV)-NCAVS(1)+1
      NXIN=RNC*RNG(1)
      NSS=NCAVS(1)+NXIN
      T2= FLOATF(NSS)*33.766666
C DETERMINE MIN. DEPTH FROM CAVITATION DEPTH VS. SPEED TABLE
      DO 6 I=1,NCAV
      IF(NSS-NCAVS(I)) 5,5,6
    5 RNC=NCAVD(NCAV)-NCAVD(I)+1
      NXIN=RNC*RNG(1)
      DT= NCAVD(I)+ NXIN
      GO TO 10
    6 CONTINUE
C
C  COMPUTE VELOCITY COMPONENTS OF SUBMARINE AND HELICOPTER
C   DETECTION RANGES.
   10 VXS=T2*COSF(T1)
      VYS=T2*SINF(T1)
      CALL COMPRG
      GO TO 115
C  IF TRACK IS PREDETERMINED ASK IF FIRST COURSE IS RANDOM
  102 STIME = 0.0
      IF (NRANC)103,103,110
  103 RNC=360
      NXIN=RNC*RNG(1)
      TT=NXIN
      GO TO 110
  108 IF(SCUS(KT)-SCUS(KT-1))110,109,110
  109 IF(SSPD(KT)-SSPD(KT-1))110,112,110
  110 SPD=SSPD(KT)*33.7666666
      CUS=(-SCUS(KT)+TT)/57.29578
      XS=XS+(SMTIME(KT)-STIME)*VXS
      YS=YS+(SMTIME(KT)-STIME)*VYS
      VXS=SPD*COSF(CUS)
      VYS=SPD*SINF(CUS)
      STIME = SMTIME(KT)
      IF(DEPTH(KT)-DT) 112,12,112
  112 DT = DEPTH(KT)
      CALL COMPRG
   12 IF(KT-NRMAN)113,114,114
C  IF THIS IS NOT THE LAST MANEUVER STORE NEW EVENT 1 AT
C   NEXT MANEUVER TIME.
  113 KT=KT+1
      TIMET=SMTIME(KT)
      NREVT = 1
      SMNEXT = TIMET
      CALL SNE
      GO TO 115
  114 TT=0.0
      SMNEXT= 12345.0
  115 RETURN
      END
```

```
      SUBROUTINE HDE
C
C  HELICOPTER DETECTION EVENT
C
C           *** DIMENSION AND COMMON STATEMENTS ***
C
      I = NRUNT
      IF(NSWP(I)-1) 300,300,302
  300 IDIP=IDIP+1
  302 T1=TIMET
      T2 = T1-STIME
C
      J1=NSWP(I)
      DO 324 J=J1,NRSW
      XT = XS + VXS* T2
      YT = YS + VYS* T2
      RTT = SQRTF(( XH(I) - XT)**2 +(YH(I)-YT )** 2 )
      IF(RTT-DRNG(I,J)) 310,310,319
  310 IF(RTT - SHORTR ) 319,319,311
  311 NOVER = 1
      IDIP=0
      NDET =NDET + 1
      T1      = T1 + SWT(J) - DAT
      GO TO 34C
  319 T1=T1+SWT(J)
      T2=T2+SWT(J)
C DID SUBMARINE MANEUVER DURING THIS SWEEP. IF ANSWER IS YES
C  ASK IF THIS IS THE LAST SWEEP.
      IF(SMNEXT -T1) 320,320,324
  320 IF(J-NRSW) 321,324,324
C  RESTORE THIS EVENT TO ALLOW SUBMARINE TO MANEUVER BEFORE
C    COMPLETING SONAR SEARCH.
  321 NSWP(I)=J+1
      TIMET=T1
      NREVT=3
      CALL SNE
      GO TO 34C
C
  324 CONTINUE
I     IF(IDIPN(I)-NRDIPS)329,325,325
  325 IF(    IDIP- I2 ) 340,326,326
  326 NOVER = 1
      IDIP=C
      GO TO 34C
  329 IF(FCPR) 334,334,330
  330 FCT=RNG(1)
      IF(FCT-FCPR) 332,332,334
C
  332 FCT=(FCT/FCPR)*FCTMAX
      T1 = T1 +FCT
  334 TIMET = T1+TRD
C
      NREVT = 2
      CALL SNE
      NSWP(I)= 1
  340 RETURN
      END
C
      SUBROUTINE PRINT
C
      DIMENSION  NR(10)
C           *** DIMENSION AND COMMON STATEMENTS ***
C
      DO 5000 I = 1,10
 5000 NR(I) = I
C
   10 READ INPUT TAPE 5, 1000,NAME,N1,N2,N3,N4,N5
 1000 FORMAT(A6,5I10)
   19 IF(NAME - 4HDATE) 25,20,25
   20 NDAY = N1
      MONTH =N2
      NYEAR =N3
```

```
      J1=1
      J2 = 0
      NRU  =1
      KDET = 0
   11 WRITE OUTPUT TAPE 6,3000,MONTH,NDAY,NYEAR
 3000 FORMAT(1H1,34X,36H AHS-1 AN ASW HELICOPTER SIMULATION /
     1/ 10X,5H DATE I2,1H/ I2,1H/ I2 ///)
      WRITE OUTPUT TAPE 6,3001,NRU
 3001 FORMAT (38X,26H INPUT DATA FOR RUN NUMBER ,I3 ///)
      GO TO 10
C
   25 IF(NAME - 5HNRSAM ) 27,26,27
   26 IF(KDET) 206,206,207
  206 NRSAM = N1
      KDET = N2
      IRNC=N3
C
      READ INPUT TAPE 5,1003, (NSSIZE(I),I=1,NRSAM )
 1003 FORMAT (1216)
      GO TO 10
C
  207 IF(J1-J2 ) 209,208,208
  208 WRITE OUTPUT TAPE 6,2040,J1
 2040 FORMAT( //  9X,36H COMPARATIVE DETECTION DATA FOR RUN
     1,6HNUMBER,I3///)
      J1 = NRU
      KDET = 0
C
      GO TO 225
  209 WRITE OUTPUT TAPE 6,2041,J1,J2
 2041 FORMAT(  // 9X, 36H COMPARATIVE DETECTION DATA FOR RUNS
     1,I3,8H THROUGH,I3 //)
      J1 = NRU
      KDET = 0
      GO TO 225
C
  225 T2=JT
      BARD = BARD/T2
      VARD = (VARD - T2* BARD**2)/(T2-1.)
C
      WRITE OUTPUT TAPE 6,2042,BARD,VARD,MIND,MAXD
 2042 FORMAT(/// 9X,32H NUMBER OF DETECTIONS PER SAMPLE //
     1 9X,18H SAMPLE MEAN      = F13.4 /9X,18H SAMPLE VARIANCE =
     2   F13.4 /9X,18H MINIMUM          = I8/9X, 8H MAXIMUM,
     39X,1H= I8 )
C
      BARD=0.0
      VARD=0.0
      MIND=4321
      MAXD=0
      JT=0
  229 IF(K1) 206,206,411
   27 IF(NAME - 6HNHISPD ) 30,28,30
   28 NHISPD = N1
      LOSPD = N2
      MAXSD = N3
      MINSD = N4
      GO TO 10
C
   30 IF(NAME - 6HNRANSS) 40,31,40
   31 NRANSS = N1
      NRANC = N2
      GO TO 10
C
   40 IF(NAME - 5HNRMAN ) 60,41,60
   41 NRMAN = N1
      READ INPUT TAPE 5, 1001,(  SMTIME(I),DEPTH(I),SCUS(I),
     1SSPD(I), I=1,NRMAN )
 1001 FORMAT(4F10.5)
      GO TO 10
```

```
   60 IF(NAME - 4HNRSW ) 65,61,65
   61 NRSW = N1
      READ INPUT TAPE 5, 1002,(SWT(I),PD(I), I = 1,NRSW )
      GO TO 10
C
   65 IF(NAME - 3HTOD) 68,66,68
   66 READ INPUT TAPE 5,1006,TOD,TRD,FCPR,FCTMAX
 1006 FORMAT(4F10.5)
C
      GO TO 10
C
   68 IF(NAME - 4HNRHS) 70,69,70
   69 NRHS = N1
      NRDIPS =N2
      HSPD=N3
      NTE=N4
C
      DO 5 I=1,NRHS
   52 READ INPUT TAPE 5, 1002,(HBRG(I,J),HTE(I,J),J=1,NRDIPS )
 1002 FORMAT (6F10.5)
C
CONVERT HELO DISTANCES TO TIME ENROUTE AND DEGREES TO RADIANS
      HBRG(I,1)= (-HBRG(I,1)+90.0)/57.29578
      DO 5 J=2,NRDIPS
    5 HBRG(I,J)=(-HBRG(I,J)/57.29578)+HBRG(I,J-1)
      HSPDT = HSPD* 33.76666666
      IF(NTE) 53,53,55
   53 DO 54 I=1,NRHS
      DO 54 J=1,NRDIPS
   54 HTE(I,J) = HTE(I,J)/HSPDT
   55 GO TO 10
C
   70 IF(NAME - 4HXDAT ) 80,71,80
   71 XDAT=N1
      YDAT = N2
      GO TO 10
C
   80 IF (NAME -5HSONAR ) 90,81,90
   81 NS=N1
      READ INPUT TAPE 5,1005,FOMMU,FOMVAR,DL,T,F,TS,PCL
 1005 FORMAT( 7F10.5)
      GO TO 10
C
   90 IF(NAME - 3HDAT) 100,91,100
   91 READ INPUT TAPE 5, 1008,DAT,NAME
 1008 FORMAT (  F10.5,A6 )
      IF (NRU  - 1) 93,93,92
   92 WRITE OUTPUT TAPE 6, 2001,NRU
 2001 FORMAT (1H1,////45X,26H INPUT DATA FOR RUN NUMBER I3 ///)
   93 WRITE OUTPUT TAPE 6,2002,NAME,NRHS,DAT
 2002 FORMAT(26X,12HSEARCH PLAN ,A6,6H WITH I2, 9H HELICOPT
     1,8HERS AND ,F3.0,18H MINUTES TIME LATE // )
      GO TO 10
C
C
  100 IF(NAME-4HNCAV) 110,102,110
  102 NCAV=N1
      READ INPUT TAPE 5,999, (NCAVS(I),NCAVD(I),I=1,NCAV)
  999 FORMAT(12I6)
      NHISPD = NCAVS(NCAV)
      LOSPD =NCAVS(1)
      MAXSD = NCAVD(NCAV)
      MINSD=NCAVD(1)
      GO TO 10
C
  110 IF(NAME-2HGO) 311,330,311
C
  311 IF (NAME - 5HFINIS) 312,313,312
  312 WRITE OUTPUT TAPE 6,2000,NAME
 2000 FORMAT (1H1,31X,36HFOLLOWING DATA CARD CONTAINS ILLEGAL
     1,5H NAME //12X,A6,42X,12HJOB DELETED. )
      STOP
```

```
  313 K1=1
      GO TO 207
C PRINT OUT HEADING AND INPUTS
C
  330 WRITE OUTPUT TAPE 6,2010,HSPD,TDD,FOMMU,TRD,FOMVAR,
     1NRDIPS,F,NRSW,NS
 2010 FORMAT (/ 25X,15HHELICOPTER DATA,35X,11H SONAR DATA //
     19X,11H JUMP SPEED,17X,3H = F5.0,5HKNOTS,25X,
     217H FIGURE OF MERIT //10X,25HTIME TO LOWER TRANSDUCER
     3,3H = F5.1,8H MINUTES,24X,12HAVERAGE    = F8.3 /10X,
     429HTIME TO RAISE TRANSDUCER    = F5.1,8H MINUTES,24X,
     512HVARIANCE    = F8.3 //10X,24HNUMBER DIPS EACH HELICOP
     6,5HTER = I4,31X,23H ACOUSTIC FREQUENCY    = F6.2,6H  KC.
     7//10X,29HNUMBER SWEEPS EACH DIP       = I4,31X,5H SEA ,
     816HSTATE              = I4/ )
      WRITE OUTPUT TAPE 6,2012,DL,T,(I,PD(I),SWT(I),I=1,NRSW)
 2012 FORMAT(10X,5HSONAR,7X,5HDEPTH,6X,6H SWEEP,35X,
     122H LAYER DEPTH          = F6.2,6H  FEET /10X,5HSWEEP,
     220X,4HTIMES,35X,23H TEMPERATURE IN LAYER = F6.1,4H  F.
     3//(10X,I3,F14.0,F12.1 /// )
C
      WRITE OUTPUT TAPE 6,2011,FCPR,FCTMAX
 2011 FORMAT(/9X,31H PROBABILITY OF FALSE CONTACT = ,F6.5 /9X,
     131H MAX. DELAY FOR FALSE CONTACT = F5.1,8H MINUTES / )
  380 IF(NRANSS)382,382,401
  382 IF(NCAV) 400,400,384
C
  384 WRITE OUTPUT TAPE 6,2018
 2018 FORMAT(//47X, 25H NON-CAVITATING SUBMARINE)
C
  400 WRITE OUTPUT TAPE 6,2020,LOSPD,NHISPD,MINSD,MAXSD
 2020 FORMAT(/35X,40HSUBMARINE COURSE, SPEED AND DEPTH CHOSEN
     1, 9H RANDOMLY//10X,16HMINIMUM SPEED  = I4 /10X,
     216HMAXIMUM SPEED  = I4 //1/X,16HMINIMUM DEPTH  = I5 /
     310X,16HMAXIMUM DEPTH  = I5 )
      GO TO 405
C
  401 IF(NRANC) 402,402,403
  402 WRITE OUTPUT TAPE 6,2021
 2021 FORMAT( 34X, 37HSUBMARINE TRACK PRESET(FIRST COURSE
     1,15HCHOSEN RANDOMLY  // )
      GO TO 404
C
  403 WRITE OUTPUT TAPE 6,2022
 2022 FORMAT( 49X,22HSUBMARINE TRACK PRESET // )
  404 WRITE OUTPUT TAPE 6,2023, ( SMTIME(I),SCUS(I),SSPD(I),
     1DEPTH(I),I=1,NRMAN )
 2023 FORMAT(10X,4HTIME  ,6X, 7H COURSE,6X,14H SPEED (KNOTS),
     16X,13H DEPTH (FEET) //(10X,F6.1,F11.0,F16.0,F22.0 ))
C
  405 WRITE OUTPUT TAPE 6,2030,NRU
 2030 FORMAT(     ///45X,26HOUTPUT DATA FOR RUN NUMBER I3 //)
      WRITE OUTPUT TAPE 6,2031,(NR(I),I=1,NRHS)
 2031 FORMAT(///  30X,10H SUBMARINE,10X,17H  SEARCH TIME TO
     1,10HDETECTION ,7X,20HNUMBER OF DETECTIONS /30X,
     210HDETECTIONS,10X,25H  WHEN SUBMARINE DETECTED 10X,
     320H BY EACH HELICOPTER  /59X,9H(MINUTES) /10X,
     415HSAMPLE   SAMPLE,21X,5H MEAN,14X,17HMINIMUM   MAXIMUM
     5/10X,44HNUMBER    SIZE     NUMBER   PERCENT   TIME  ,
     628HVARIANCE    TIME        TIME  ,10I4 )
C
      WRITE OUTPUT TAPE 6,2032
 2032 FORMAT(//)
C
      NRU  = NRU  + 1
      J2 = J2 + 1
C
  410 RETURN
  411 STOP
      END
```

```
      SUBROUTINE COMPRG
C
C          *** DIMENSION AND COMMON STATEMENTS ***
C
      DO 220 J=1,NRSW
      DP = PD(J)
      CALL SETCOF
C
      DO 220 I = 1,NRHS
      PLG=PL(I)
      IF(DL) 204,204,202
  202 IF(PLG - P(1)) 203,203,210
  203 CALL   SCR(1)
      DRNG(I,J) = RG*1000.0
      IF(DT - DL) 22,22,204
   22 IF(DP - DL) 220,204,204
  204 IF (RC) 27,27,24
   24 IF(PLG- FL* LOGF(RG) - B(5)*RG - C(5)) 220,220,206
  206 CALL SCR(5)
      DRNG(I,J) = RG*1000.0
      GO TO 220
   27 IF(DL) 208,208,207
  207 IF(PLG- A(4)*FLN*LOGF(RG)-B(4)*RG - C(4)) 220,220,208
  208 CALL SCR(4)
      DRNG(I,J) = RG*1000.0
      GO TO 220
  210 IF(PLG- P(2)) 211,211,212
  211 CALL SCR(2)
      DRNG(I,J) = RG*1000.0
      GO TO 204
  212 CALL SCR(3)
      DRNG(I,J) = RG*1000.0
      GO TO 204
  220 CONTINUE
      RETURN
      END
C
      SUBROUTINE SCR(KP)
C          *** DIMENSION AND COMMON STATEMENTS ***
      IF (KP - 3)     110,111,110
  110 C2 = 6.0206
      GO TO 10
  111 C2 = 3.0103
   10 C0 = B(KP)
      C1 = C(KP)
      IF (PLG-C0-C1)  14,11,11
   11 C4= C0
      C5= C2 + 2.*C4 + C1
      C6= 0.
      C7= 1.
   12 IF (C5-PLG)  13,17,17
   13 C4 = 2.*C4
      C5 = C2 + C4 + C5
      C6 = C6 + C7
      GO TO 12
   14 C4 = C0*.5
      C5 = C1 + C4 - C2
      C6 = -1.
      C7 =-1.
   15 IF (PLG-C5)  16,17,17
   16 C4 = C4*.5
      C5 = C5 - C2 - C4
      C6 = C6 + C7
      GO TO 15
   17 TOP = PLG - C1 - C2*(C6-.95693)
      DENOM = C0 + C2*(.5**C6)
      C12 = TOP/DENOM
      C15 = (1.44269)*C2
   18 T1 =  C15* LOGF  (C12) + C0*C12 + C1 - PLG
      T2 = (C15/C12) + C0
```

```
      C13 = C12 - (T1/T2)
      IF (C13)  25,25,19
   19 IF (ABSF(C13-C12)-.001)    21,21,20
   21 RG = C13
      GO TO 50
   20 C12 = C13
      GO TO 18
   25 WRITE OUTPUT TAPE 6,26
   26 FORMAT(1X,40H PROGRAM DETECTED AN ERROR - NEGATIVE OR,
     118HZERO RANGES IN SCR)
      CALL EXIT
   50 RETURN
      END
C
      SUBROUTINE SETCOF
C          *** DIMENSION AND COMMON STATEMENTS ***
  111 IBIT=2
      A6=(F/25.)**0.3333333333
      A7=(1.23)*(10.**(6.-(2100.)/(T+459.6)))
      A8=F*F
      A10=SQRTF(ABSF(DP-DL)) + SQRTF(ABSF(DT-DL))
      A16=(0.65)*A7*A8)/(A8+A7*A7)+(0.0269*A8)/A7
      IF (DL) 133,133,112
  112 A0=SQRTF(DL)
      A1=SQRTF(DP)
      A2=A1/A0
      A3=SQRTF(DT)
      A4=A3/A0
      A5=(4.5/A0)*SQRTF(F)
      IF (F-8.)  113,114,114
  113 A9=1.
      GO TO 115
  114 A9=1.462*A6
  115 IF (1.-A2)  116,119,119
  116 IF (1.-A4)  117,118,118
  117 A11=0.2*(SQRTF(A2*A2-1.) + SQRTF(A4*A4-1.))
      GO TO 122
  118 A11=(1.-A4)*.25 + .2*(SQRTF(A2*A2-1.))
      GO TO 122
  119 IF (1.-A4)    121,120,120
  120 A11=(2.-A2-A4)*0.25
       IF (A2-A4) 122,200,122
  200  A12 = FL*(-6.9)
      GO TO 12
  121 A11     = (1.-A2)*0.25+0.2*(SQRTF(A4*A4-1.))
  122 A12= FL*LOGF(A0*A11)
   12 A13= FL*LOGF((A0)*(A11+.5))
      A18=ABSF(A2-A4)
  123 A15= .4*A9*((10.**A2)+(10.**A4)+(10.**A18))
      IF (A18-1.)  124,125,125
  124 A14 = .1*A6*((10.)**(2.3*A18))
      GO TO 126
  125 A14 = 20.*A6
  126 IF (NS-3)  127,128,128
  127 A17 = A5
      GO TO 129
  128 A17 = 2.*A5
  129 A19 = 2.*(A15-A14)
      A20 = A11*A19
      A27 = A16 + (A19/A0)
      A28 = A14 + 60.
      A25 = A28 + A12 + A27*A0*A11 - A20
      A26 = A28 + A13 + A27*A0*(A11+.5) - A20
      IF  (DL)  133,133,130
  130 A(1)=20.
      B(1)= A16 + (A14/(A0*A11))
      C(1)= 60.
      P(1)= A25
```

```
  131 A(2)= 20.
      B(2)= A16 + (A19/A0)
      C(2)= A28 - A20
      P(2)= A26
  132 A(3)= 10.
      B(3)= A16 + A17
      C(3)= A15 - A17*A0*(A11+.5) + .5*A13 + 60.
      P(3)= A26
  133 A(4)= 20.
      B(4)= A16 + 5.*A6
      C(4)= (25.-A10)*A6 + 60.
      RC  = (A10-25.)*.2
      IF (RC)  33,33,34
   33 PLC=123456.
      P(4)=PLC
      GO TO 134
   34 PLC=  FL*LOGF(RC) +A16*RC + 60.
      P(4) = PLC
  134 A(5) = 20.
      B(5) = A16
      C(5) = 60.
       P(5) = PLC
      RZ1 =  A0*A11
      RZ2 =  A0*(A11+.5)
      IF (RZ1-.01)  135,135,136
  135 P(1)=-2468.
  136 RETURN
      END
C
      SUBROUTINE NORM
C
C          *** DIMENSION AND COMMON STATEMENTS ***

      YNORM = C.0
      DO 10 IN = 1,12
   10 YNORM=YNORM+RNG(1)
      YNORM = XMU+( YNORM - 6.0 ) * SQRTF ( SIG2)
      RETURN
      END
      FUNCTION RNG(NR)
C
C UNIFORM (0,1) FLOATING POINT NUMBER GENERATOR. (NR,0 -RESET
C,NR=1 - GENERATE NEXT NUMBER IN THE SEQUENCE AND PLACE IN A
C REGISTER,NR=2 OR MORE- CYCLE GENERATOR NR TIMES)
C NUMBERS GENERATED BY POWER RESIDUAL METHOD.
C
      CON ( K1=1220703125   , K2=2000000000000000B,  R=777777777B,
     1 R1=77777777 B)
C
      IF(NR) 10,10,20
   10 R=R1
      GO TO 40
   20 DO 30 I=1,NR
      LDA(R),MUF(K1),DVF(K2),STQ(R)
   30 CONTINUE
      LLS(48),DVF(K2),ARS(11),ADD(K2),FAD(K2)
   40 RETURN
      END
      END
```

INPUT DATA FOR RUN NUMBER 4

SEARCH PLAN A131 WITH 4 HELICOPTERS AND 15. MINUTES TIME LATE

HELICOPTER DATA

JUMP SPEED = 80.KNOTS
TIME TO LOWER TRANSDUCER = 1.5 MINUTES
TIME TO RAISE TRANSDUCER = 1.5 MINUTES

NUMBER DIPS EACH HELICOPTER = 5
NUMBER SWEEPS EACH DIP = 2

SONAR SWEEP	DEPTH	SWEEP TIME
1	50.	1.0
2	150.	1.5

PROBABILITY OF FALSE CONTACT =.30000
MAX. DELAY FOR FALSE CONTACT = 20.0 MINUTES

SONAR DATA

FIGURE OF MERIT
AVERAGE = .000
VARIANCE = 100.000

ACOUSTIC FREQUENCY = 20.00 KC.
SEA STATE = 2
LAYER DEPTH =100.00 FEET
TEMPERATURE IN LAYER = 50.0 F.

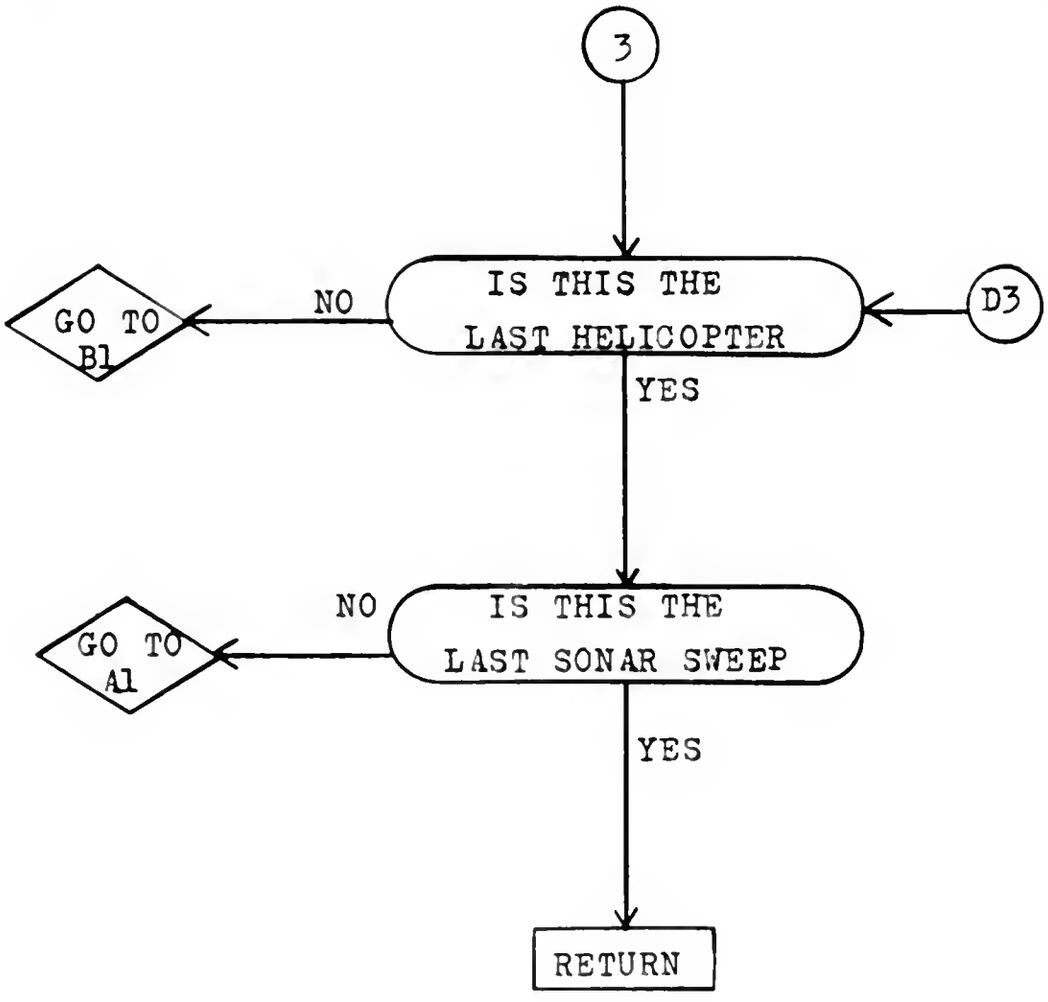
SUBMARINE COURSE, SPEED AND DEPTH CHOSEN RANDOMLY

MINIMUM SPEED = 5
MAXIMUM SPEED = 15

MINIMUM DEPTH = 100
MAXIMUM DEPTH = 500

OUTPUT DATA FOR RUN NUMBER 4

SAMPLE NUMBER	SAMPLE SIZE	SUBMARINE DETECTIONS NUMBER	SUBMARINE DETECTIONS PERCENT	SEARCH TIME TO DETECTION WHEN SUBMARINE DETECTED (MINUTES) MEAN TIME	VARIANCE	MINIMUM TIME	MAXIMUM TIME	NUMBER OF DETECTIONS BY EACH HELICOPTER 1	2	3	4
1	100	46	46.000	6.72	27.1239	2.50	23.13	9	11	12	14


3
IS THIS THE LAST HELICOPTER
NO
GO TO B1
D3
YES
IS THIS THE LAST SONAR SWEEP
NO
GO TO A1
YES
RETURN

Subroutine NORM. The function of Subroutine NORM is to compute the value of a Normal random variable, employing the classical central limit theorem. Given a sequence of independent, identically distributed random variables X_i with finite mean $E(X)$ and standard deviation $\sigma(X)$, then the sequence Y_i defined by

$$Y_n = \frac{(X_1 + X_2 + \cdots + X_n) - nE(X)}{\sqrt{n}\,\sigma(X)} \tag{6.9}$$

converges in distribution to a random variable which is normally distributed with mean zero and variance one. [27]

In actual practice the value of a pseudo-normally distributed random variable is computed since pseudo-random numbers generated by RNG are used for the sequence X_i. Under the assumption that the X_i are uniformly distributed in the interval $[0,1]$ then $E(X) = \frac{1}{2}$ and

$$\sigma(X) = \frac{1}{\sqrt{12}}$$

Taking n as twelve we compute

$$Y = \frac{\sqrt{12}}{\sqrt{12}} \left[(X_1 + X_2 + \cdots + X_{12}) - 6\right] \tag{6.10}$$

which is approximately normal with mean zero and variance one. To obtain the value of a normal random variable Z, with mean μ and variance σ^2 we use the transformation

$$Y = \frac{Z - \mu}{\sigma} \tag{6.11}$$

or

$$Z = \mu + \sigma Y$$

The following calling sequence is used for Subroutine NORM:

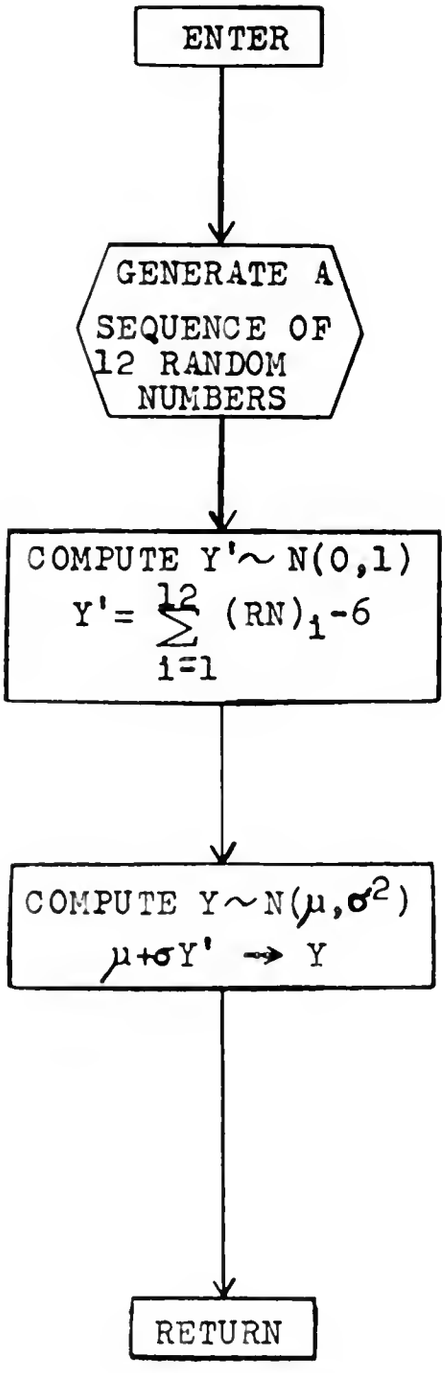
$\mu \rightarrow$ XMU

$\sigma^2 \rightarrow$ SIG2

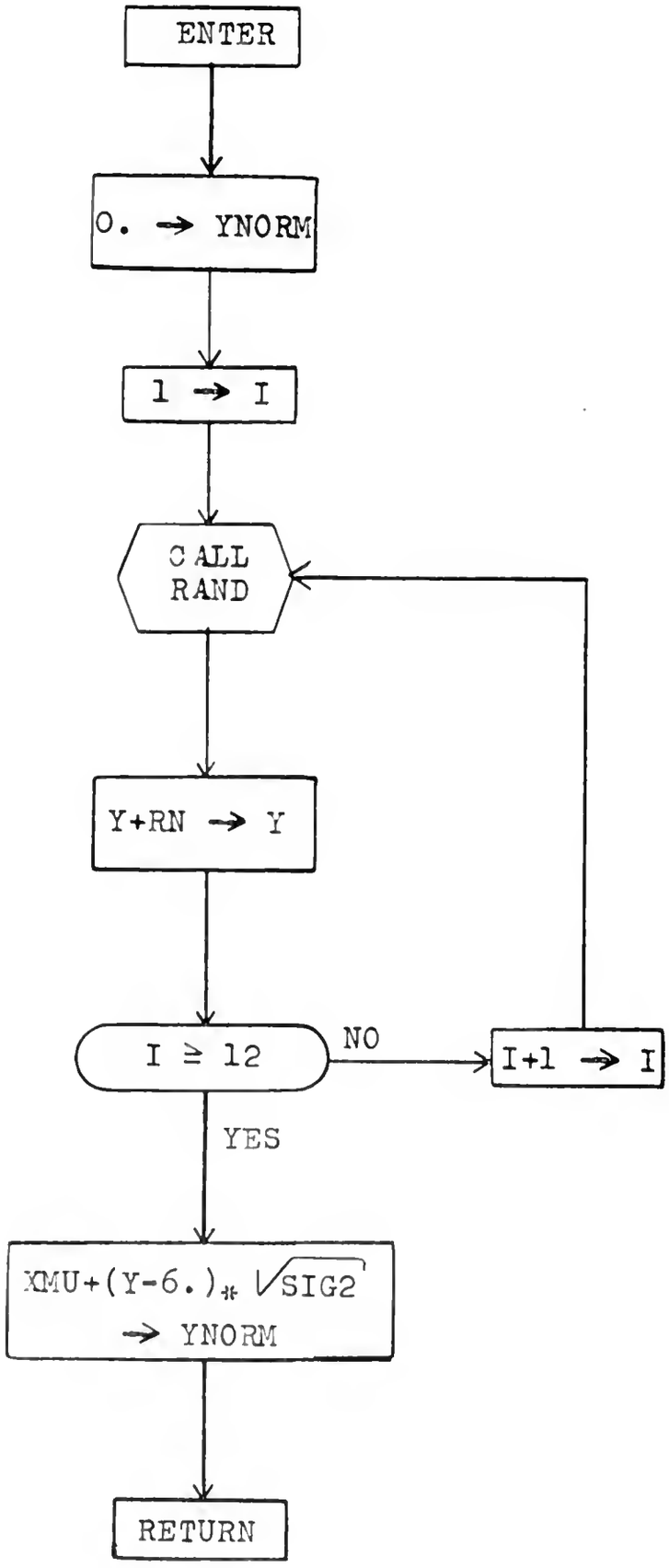
CALL NORM

The resulting normal variate will be placed into the variable YNORM by the subroutine.

Subroutine NORM (General)



Subroutine NORM (Detailed)



BIBLIOGRAPHY

References Cited

1. U. S. Naval War College. Fundamentals of War Gaming, by F. J. McHugh. 1961.

2. Johns Hopkins University, Operations Research Office. A Survey of Historical Developments in War Games, by J. P. Young. 1959.

3. Pennington, A. W. History and Classification of War Games. First War Gaming Symposium Proceedings, J. L. Overholt, editor. Washington Operations Research Council, 1962.

4. U. S. Department of Commerce, National Bureau of Standards. Monte Carlo Method. A. S. Householder, editor. Applied Math. Series No. 12, 1951.

5. Paxson, E. W. War Gaming. RAND Corp. RM - 3489-PR, 1963.

6. Meyer, H. A. (ed.). Symposium on Monte Carlo Methods, University of Florida, 1954. Wiley, 1956.

7. Technical Operations Inc. The Game of War. 1960.

8. Adams, R. H. and J. L. Jenkins. Simulation of Air Operations with the Air Battle Model. JORSA, v. 8, No. 5, October 1960.

9. Daniels, A. E. The User's Viewpoint on the Creation and Application of a Large Simulation Model. First War Gaming Symposium Procedings. J. L. Overholt, editor. Wash. Ops. Research Council, 1962.

10. Wagner, I. F. An Introduction to the Design and Analysis of War Games. Planning Analysis Group, Applied Physics Laboratory, Johns Hopkins Univ., 1962.

11. Davies, O. L. (ed.). The Design and Analysis of Industrial Experiments. Hafner Pub. Co., 1956.

12. Cox, D. R. Planning of Experiments. Wiley, 1958.

13. Thomas, C. J. Progress in Operations Research. R. L. Ackoff, editor, v. 1, JORSA No. 5, Wiley, 1961.

14. Adams, H. E. Machine-Played Games. First War Gaming Symposium Proceedings. Wash. Ops. Research Council, 1962.

15. Galliher, H. P. Simulation of Random Processes. Notes on Operations Research, 1959. The Technology Press, 1959.

16. Harling, John. Simulation Techniques in Operations Research - A Review. JORSA, v. 6, No. 3, (May-June, 1958).

17. Klinkner, R. L. A Sonar Detection Scheme for Computer War Games. Planning Analysis Group, Johns Hopkins Univ. PAM - 67, 1963.

18. Van Nostrand's Scientific Encyclopedia, 3rd ed. D. Van Nostrand Co., 1958.

19. Horton, J. W. Fundamentals of SONAR. United States Naval Institute, 1959.

20. Redgrave, M. J. Some Approaches to Simulation, Modeling, and Gaming at SDC. Systems Development Corp. SP - 721, March 19, 1962.

21. Saaty, T. L. Mathematical Methods of Operations Research. McGraw-Hill, 1959.

22. Pennington, A. W. War Game Techniques. Paper distributed at the 5th SHAPE OR/ Scientific Advisory Conference. 18-20 May, 1960.

23. Jenkins, J. L. A Collection of Simulation Models for the Study of Air War. Paper given at the 19th National Meeting of ORSA, 26 May, 1961.

24. Markowitz, H. M., Bernard Hauser and H. W. Karr. SIMSCRIPT A Simulation Programming Language. RAND Corp., Prentice-Hall, 1963.

25. Systems Research Group Incorporated. A Preview of MILITRAN - a System for the Development of Computer Programs that Simulate Military Systems. January, 1962.

26. Matteis, R. J. and W. C. Suhler. An Approach to Sensitivity Analysis of Carmonette. Proceedings of the seventh conference on the design of experiments in army research development and testing, ARODR 62-2, 1962.

27. Parzens, Emanuel. Modern Probability Theory and Its Applications. Wiley, 1960.

Selected General Bibliography

Biser, E. and M. Meyerson. The application of Design of Experiments and Modeling to Complex Weapons Systems. JORSA, v. 5, No. 2, April 1957.

Clark, C. E. Importance Sampling in Monte Carlo Analysis. Systems Development Corp., TM - 505, June 24, 1960.

Jacoby, Joan E. and Stephen Harrison. Efficient Experimentation with Simulation Models, Technical Operations, Inc., OMEGA Tr 60-2, June 1960.

Kahn, Herman and Irwin Mann. Monte Carlo. RAND Corp., P - 1165, July 30, 1957.

Livermore, W. R. The American Kriegspiel. Houghton Mifflin and Co., The Riverside Press, 1882.

Overholt, John (ed.). First War Gaming Symposium Proceedings. Washington Operations Research Council, 1961.

Sayre, Farrand. Map Maneuvers and Tactical Rides. Army Service Schools Press, 1910.

Thomas, C. J. and W. L. Deemer, Jr. The Role of Operational Gaming in Operations Research. JORSA, v. 5, No. 1, Feb. 1957.

Verdy du Vernois, Julius A. F. W. von. Simplified War Game. Hudson - Kimberly, 1897.

Wagner, I. F. A Study in Complete Factorial and Fractional Factorial Game Designs Using the AAW Mod 0. Planning Analysis Group, APL, Johns Hopkins Univ., PAIN - 29, June 29, 1961.

Weiner, M. G. War Gaming Methodology. RAND. RM - 2413, 1959.

APPENDIX A

LISTING OF AHS-1 FORTRAN COMPUTER CODE

```
      PROGRAM AHS 1
C
      DIMENSION  NSSIZE(100),NHSDET(10),HTE(10,20),HBRG(10,20)
     1,XH(10),YH(10),XHT(10,20),YHT(10,20),IDIPN(10),SMTIME(20)
     2,DEPTH(20),SCUS(20),SSPD(20),PD(5),TIME(31),NREV(31),
     3NRUN(31),DRNG(10,5),SWT(5),NSWP(10),NCAVD(30),NCAVS(30)
     4,A(10),B(10),C(10),P(10),PL(10)
      COMMON IT,JT,T1,T2,IDIP,I2,TT
      COMMON NRSAM,NSSIZE,NHSDET,TMIN,TMAX,BART,VART,MIND,
     1MAXD,BARC,VARD,PERC,DAT,XDAT,YDAT,NDET,NDTEMP,PL
      COMMON TIMET,NREVT,NRUNT,NTNE,TIME,NREV,NRUN
      COMMON KT,NRANSS,NRANC,STIME,DT,VXS,VYS,YS,XS,SMTIME,
     1SCUS,SSPD,NRMAN,NHISPD,LOSPD,MAXSD,MINSD,NCAV,NCAVS,
     2NCAVD,DEPTH
      COMMON IDIPN,NRDIPS,HTE,HBRG,XH,YH,NRHS,HSPD,HSPDT,TDD
     1,XHT,YHT,NTE
      COMMON DRNG,NRSW,SWT,PD,TRD,NOVER,SHORTR,SMNEXT,NSWP
      COMMON FCPR,FCTMAX,FCT
      COMMON NS,T,F,DL,DP,TS,FOMMU,FOMVAR
      COMMON A,B,C,P,PLG,RG,AR,FL,FLN,FL3
      COMMON IBIT,IFS,IR,PLC,RC,RS,RT,RZ1,RZ2,SG
      COMMON A0,A1,A2,A3,A4,A5,A6,A7,A8,A9,A10,A11,A12,A13,
     1A17,A18,A19,A20,A21,A22,A23,A24,A25,A26,A27,A28,A29,A30
     2,A14,A15,A16
      COMMON C3,C1,C2,C4,C5,C6,C7,C12,C13,C15,DENOM,TOP
      COMMON YNORM,SIG2,XMU
C
      XDAT=0.0
      YDAT=0.0
      IRNC=0
      NCAV = 0
      FCT=0.0
      FCPR=0.0
      JT=0
      BARD = 0.0
      VARD = 0.0
      MIND = 1000
      MAXD = 0
      FLN =.4342944819
      FL = 20.*FLN
      FL3 = 10.*FLN
    1 STIME = 0.0
      NRANC = 0
      NRANSS = 0
      CALL PRINT
      SMNEXT = 12345.0
      TT=90.
C
C   TEST NTE TO DETERMINE IF HELO BEARING AND TIMING ERRORS
C   ARE TO BE COMPUTED. IF NOT,COMPUTE DIP STATIONS.
      IF(NTE) 11,11,12
   11 DO 12 I=1,NRHS
      T3=XDAT
      T4=YDAT
      DO 12  J=1,NRDIPS
      T1 = HBRG(I,J)
      T2 = HTE(I,J)
      VXH = HSPDT *COSF(T1)
      VYH = HSPDT *SINF(T1)
      T3 = T3+VXH*T2
      T4 = T4+VYH*T2
      XHT(I,J)=T3
      YHT(I,J)=T4
   12 CONTINUE
```

NOTE. DIMENSION AND COMMON STATEMENTS ARE IDENTICAL FOR ALL SUBROUTINES AND ARE LISTED ONLY IN THE MAIN PROGRAM

```
      JT=JT+NRSAM
      XMU = FOMMU
      SIG2=FOMVAR
      TIME(1) = 0.0
      I2= NRHS*NRDIPS
      IDIP=0
C  CYCLE RANDOM NUMBER GENERATOR
      T3=RNG(0)
      T3=RNG(IRNC)
C
      DO 30 N=1,NRSAM
      DO 6 I =1,NRHS
    6 NHSDET(I) = 0
      BART =0.0
      VART =0.0
      TMIN = 600.0
      TMAX=0.0
      NDET=0
      N5 = NSSIZE(N)
C
      DO 20 M=1,N5
      DT=-12345.
C  SET UP FIRST SUBMARINE MANEUVER
      NTNE= 1
      KT=1
      XS = XDAT
      YS = YDAT
      TIMET = 0.0
      NREVT = 1
      CALL SNE
C SET UP FIRST DIP FOR EACH HELICOPTER,INITIALIZE DIP COUNTER
      DO 7 I=1,NRHS
      IF(NTE) 16,16,15
   15 XH(I)=XDAT
      YH(I)=YDAT
   16 IDIPN(I)=0
      NSWP(I)= 1
      TIMET = CAT
      NRUNT = I
      NREVT = 2
      CALL SNE
      CALL NORM
      PL(I)= (YNORM + TS )/2.0
    7 CONTINUE
      NDTEMP =NDET
      NOVER = 0
C  TAKE FIRST EVENT.TNE WILL HAVE CONTROL UNTIL DETECTION OR
C  LAST DIP
      CALL TNE
      IF(NDTEMP-NDET) 10,20,20
   10 I = NRUNT
      NHSDET(I)=NHSDET(I) + 1
      BART = BART + T1
      TMIN = MIN1F(TMIN,T1)
      TMAX = MAX1F(TMAX,T1)
      VART = VART + T1 **2
   20 CONTINUE
      M=NSSIZE(N)
      T2 = NDET
      BART = BART/ T2
      VART = (VART - T2*BART**2) / (T2-1.0)
      PERC=(T2/FLOATF (NSSIZE(N)))*100.0
      WRITE OUTPUT TAPE 6,100,N,M,NDET,PERC ,BART,VART,TMIN,TMAX,
     1 (NHSDET(I), I=1,NRHS)
  100 FORMAT (11X,I3,7X,I3,6X,I3,3X,F7.3,F8.2,F11.4,2F10.2,2X,(9I4)
      MAXD = XMAXOF(MAXD,NDET)
      MIND = XMINOF(MIND,NDET)
      BARD = BARD + FLOATF(NDET)
      VARD = VARD + FLOATF(NDET **2)
   30 CONTINUE
      GO TO 1
      END
```